# Development and evaluation of smartphone-based ITS applications for vehicular networks

Subhadeep Patra

Advisors:

Dr. Juan-Carlos Cano Escribá
Dr. Pietro Manzoni
Dr. Peter Veelaert

*Dissertation submitted to obtain the Doctoral degree in Engineering Sciences*
*Dept. of Computer Engineering, Technical University of Valencia.*

June 18, 2019

To *my parents*, specially my mother

who never doubted me in spite of my repeated promises over the years that I would only need a couple of months more to graduate.

# Acknowledgement

Thomas Edison said, “Genius is one percent inspiration and ninety-nine percent perspiration”. This proves that even geniuses cannot do without inspiration, and it is of even more importance to an ordinary man. Hence, I would like to take this opportunity to thank all the people without whose help and support this dissertation would not have been possible to complete.

First I would like to thank my supervisors, Dr. Cano, Dr. Manzoni and Dr. Veelaert, for their valuable insights and the enormous patience that was necessary for guiding me all along the way. I am also very grateful to Professor Philips of the IPI Group, Ghent University, and Dr. González of the MindLab, Universidad Nacional de Colombia, not only for their great support during my stay at each of the campuses, but also because their expansive wealth of knowledge has given added depth to my thesis.

My sincerest gratitude to my colleagues from the Networking Research Group (GRC) of the Universidad Politécnica de Valencia, Image Processing and Interpretation (IPI) of the Ghent University, and the MindLab, Universidad Nacional de Colombia, for their valuable suggestions related to my work, but more than anything else I am indebted to them for making my stay at each of these places a very pleasant, enjoyable and memorable experience.

A special word of appreciation to the funding organisations: the *European Commission* under *Svāgata.eu*, the Erasmus Mundus Programme (EMA2); the Special Research Fund of the Ghent University; and Becas Santander: Santander Investigación 2017 for their trust and belief in my work.

Lastly, but the most important of all, my earnest gratitude and love to each and everyone of my friends: old and new that I found along the way, for their unparalleled support that kept me going even when things got really tough. And to my family members, words alone cannot describe how blessed I feel for having you guys in my life, thank you for believing in me.

*Valencia, June 18, 2019.*
*Subhadeep Patra*

# Resumen Español –Summary in Spanish–

Una de las áreas de investigación que está recibiendo más atención recientemente es la de vehículos autónomos. Los investigadores están en este momento centrados en el tercer de los cinco niveles de autonomía, los cuales son: asistencia en la conducción, automatización parcial, automatización condicional, alta automatización y automatización completa. A pesar de los rápidos progresos que están habiendo en este campo, la adopción de estas soluciones llevará tiempo no sólo debido a cuestiones legales, sino también por el hecho de que los avances tecnológicos se enfrentan a un lento respaldo por parte de los fabricantes. Además, la baja tasa de renovación de vehículos de carretera, dificulta el despliegue de tecnologías innovadoras, como es el caso de la red vehicular. Nueve años después de la introducción de la norma 802.11p para la comunicación vehicular del Instituto de Ingenieros Eléctricos y Electrónicos (IIEE), los vehículos que se usan a diario todavía carecen de la capacidad de comunicarse entre sí. Este hecho impide el uso de las muchas aplicaciones de seguridad del Sistema de Inteligencia de Transporte (SIT) que aprovecha la red vehicular para el intercambio de datos. La forma obvia de manejar este problema es poner las tecnologías disponibles a la disposición de los usuarios comunes para desarrollar soluciones que se puedan implementar fácilmente, sean cómodas de adoptar y, además, económicas.

Por esta razón, trasladamos nuestra atención a los dispositivos inteligentes, especialmente a los teléfonos inteligentes, los cuales han recorrido un largo camino desde la primera introducción de teléfonos móviles a finales del siglo XX. Hoy en día casi todos llevan uno en su bolsillo a donde sea que vayan, permitiéndoles no sólo hacer llamadas, sino también medir y controlar diferentes parámetros con la ayuda de los muchos sensores integrados que están disponibles para estos dispositivos compactos pero potentes. Nuestro objetivo es estudiar los efectos de la integración de los teléfonos inteligentes a la red vehicular para desarrollar aplicaciones de seguridad del SIT. La elección de los teléfonos inteligentes aquí no solo está justificada por su amplia disponibilidad y uso, sino también porque están evolucionando hacia terminales de alto rendimiento con microprocesadores de múltiples núcleos cargados dotados de un grupo suficientemente diverso de sensores. En esta tesis proponemos tres diferentes aplicaciones de seguridad SIT para teléfonos inteligentes, diseñados para aprovechar el entorno de red vehicular: una aplicación de generación de advertencia llamada Messiah que alerta a los conductores de la presencia de vehículos de emergencia en las cercanías; una aplicación

de Advertencia de Colisión Frontal (ACF) que advierte a los conductores si no se mantiene la distancia de seguridad mínima entre el vehículo que va delante y el que lo sigue; y, por último, una aplicación que tiene como objetivo ayudar a los conductores con asistencia visual durante el adelantamiento, llamada EYES. Todas estas aplicaciones han sido desarrolladas para la plataforma Android, y dependen de la transmisión de datos entre vehículos. Dado que los vehículos que utilizamos día a día no admiten la posibilidad de comunicarse entre sí, también diseñamos GRCBox, que es una unidad integrada de bajo coste que permite la comunicación del Vehículo a Todo (V2X).

A partir de nuestro estudio de aplicaciones para dispositivos móviles diseñados para redes vehiculares, descubrimos que el uso de teléfonos inteligentes proporciona una nueva dirección para la investigación relacionada con SIT y redes vehiculares al permitir la adopción rápida de las soluciones existentes, donde los usuarios pueden descargar y usar las aplicaciones con sólo un clic a un botón. Al mismo tiempo, la portabilidad y compacidad de los dispositivos los hace limitados en términos de velocidad, potencia de procesamiento y precisión del sensor integrado, lo que afecta al rendimiento de las aplicaciones. En nuestro caso, la aplicación más simple, Messiah, funcionó muy bien. Mientras que la aplicación EYES, dependiente de los datos del Sistema Global de Posicionamiento (GPS), y la aplicación ACF, que requería un uso intenso del procesador y de la cámara debido a su dependencia del reconocimiento de placas, se vieron afectadas por las limitaciones de hardware de los teléfonos inteligentes.

# Resum en Valencià –Summary in Valencian–

Una de les àrees d'investigació que està rebent més atenció recentment és la de vehicles autònoms. Els investigadores estan en este moment centrats en el tercer dels cinc nivells d'autonomia, els quals són: assistència en la conducció, automatització parcial, automatització condicional, alta automatització i automatització completa. Malgrat els ràpids progressos que s'estan donant en este camp, l'adopció d'estes solucions portarà temps no sols degut a qüestions legals, sinó també pel fet que els avanços tecnològics s'enfronten a un lent recolzament per part dels fabricants. A més a més, la baixa taxa de renovació de vehicles de carretera, dificulta el desplegament de tecnologies innovadores com és el cas de la xarxa vehicular. Nou anys després de la introducció de la norma 802.11p per a la comunicació vehicular de l'Institut d'Enginyers Elèctrics i Electrònics (IEEE), els vehicles que s'utilitzen a diari encara manquen de la capacitat de comunicar-se entre sí. Este fet impedeix l'ús de les moltes aplicacions de seguretat del Sistema d'Intel·ligència de Transport (SIT) que aprofita la xarxa vehicular per a l'intercanvi de dades. La forma òbvia de tractar aquest problema és posar les tecnologies disponibles a la disposició dels usuaris comuns per a desenvolupar solucions que es puguen implementar fàcilment, còmodes d'adoptar i, a més a més, econòmiques.

Per aquesta raó, traslladem la nostra atenció als dispositius intel·ligents, especialment als telèfons intel·ligents, els quals han recorregut un llarg camí des de la primera introducció de telèfons mòbils a finals del segle XX. Hui en dia quasi tots porten un en la butxaca on siga que vagen, permetent-los no sols fer cridades, sinó també mesurar i controlar diferents paràmetres amb l'ajuda dels molts sensors integrats que estan disponibles per a estos dispositius compactes però potents. El nostre objectiu és estudiar els efectes de la integració dels telèfons intel·ligents a la xarxa vehicular per a desenvolupar aplicacions de seguretat del SIT. L'elecció dels telèfons intel·ligents ací no està sols justificada per la seua àmplia disponibilitat i ús, sinó també perquè estan evolucionant cap a terminals d'alt rendiment amb microprocessadors de múltiples nuclis dotats amb un grup suficientment divers de sensors. En esta tesi proposem tres diferents aplicacions de seguretat SIT per a telèfons intel·ligents, dissenyats per a aprofitar l'entorn de xarxa vehicular: una aplicació de generació d'advertència anomenada Messiah que alerta els conductors de la presència de vehicles d'emergència en les proximitats; una aplicació Advertència de Col·lisió Frontal (ACF) que adverteix els conductors si no mantenen la distància de seguretat mínima entre el vehicle que va davant i el que

el segueix; i, per últim, una aplicació que té com objectiu ajudar els conductors amb assistència visual durant l'avançament, anomenat EYES. Totes aquestes aplicacions han sigut desenvolupades per a la plataforma Android, i depenen de la transmissió de dades entre vehicles. Donat que els vehicles que utilitzem a diari no admeten la possibilitat de comunicar-se entre sí, també dissenyem GRCBox, que és una unitat integrada de baix cost que permet la comunicació de Vechicle a Tot (V2X).

A partir del nostre estudi d'aplicacions per a dispositius mòbils dissenyats per a xarxes vehiculars, descobrim que l'ús de telèfons intel·ligents proporciona una nova direcció per a la investigació relacionada amb SIT i xarxes vehiculars al permetre l'adopció ràpida de les solucions existents, on els usuaris poden descarregar i utilitzar les aplicacions amb un sol clic a un botó. Però al mateix temps, la portabilitat i la compacitat dels dispositius els fa limitats en termes de velocitat, potència de processament i precisió del sensor integrat, cosa que afecta al rendiment de les aplicacions. En el nostre cas, l'aplicació més simple Messiah ha funcionat molt bé. Mentre que l'aplicació EYES, la qual depèn de les dades del Sistema Global de Posicionament (GPS) i l'aplicació ACF que requereix l'ús intens del processador i de la càmera degut a la seua dependència del reconeixement de plaques, s'han vist afectades per les limitacions de hardware dels telèfons intel·ligents.

# English summary

One of the research areas that is receiving a lot of attention recently is autonomous vehicles. Researchers are currently focused on the third level of autonomy out of the five levels, which are: drive assistance, partial automation, conditional automation, high automation, and full automation. Even though rapid progress is being made in this field, the adoption of these solutions will take time not only due to legal issues, but also due to the fact that technological improvements face slow endorsement by manufacturers. Also, the slow renewal rate of vehicles on road hinders the deployment of novel technologies, as is the case of Vehicular Networks (VNs). Nine years after the introduction of the Institute of Electrical and Electronics Engineers (IEEE) 802.11p standard for vehicular communication, vehicles used on a daily basis still lack the capability of communicating with one other. This fact impedes the use of the many Intelligent Transportation System (ITS) safety applications that take advantage of VNs for data exchange. The obvious way to handle this problem is to use the available technologies at the disposal of common users to develop solutions that are easily deployable, effortless to adopt, and moreover, cost effective.

For this reason we shift our attention to smart devices, specially smartphones, which have come a long way since the first introduction of mobile phones in the late 20th century. Nowadays, nearly everyone carries one in their pocket anywhere they go, allowing them to not only make calls, but also to measure and monitor different parameters with the help of the many on-board sensors that are available to these compact yet powerful devices. Our objective is to study the effects of integrating smartphones to vehicular networks, to develop ITS safety applications. The choice of smartphones here is not only justified by their wide availability and use, but also because they are evolving towards high performance terminals with multi-core microprocessors packed with a sufficiently diverse group of sensors. In this thesis we propose three different ITS safety applications for smartphones, designed to take advantage of the vehicular network environment: a warning generation application called Messiah that alerts drivers of the presence of emergency vehicles in close proximity; a Forward Collision Warning (FCW) application which warns drivers if a minimum safe distance is not maintained between the vehicle ahead and the one following it; and lastly an application that aims to aid drivers with visual assistance while overtaking, named EYES. All these applications have been developed for the Android platform, and are dependent on the data transmission among vehicles. Since vehicles we use on a day to day basis still do not accommodate the possibility to communicate with one another, we also designed the

GRCBox, which is a low cost on-board unit that supports Vehicle to Everything (V2X) communication.

From our study of applications for mobile devices designed for VNs, we found that the use of smartphones provides a new direction to research related to ITS and VNs by allowing a quick adoption of the existing solutions, where users are able to download and use applications just by one click of a button. But at the same time, the portability and compactness of the devices makes them limited in terms of speed, processing power, and accuracy of the on-board sensor, thus affecting the performance of the applications. In our case, the simpler Messiah application performed very well, while the EYES application that is dependent on Global Positioning System (GPS) data, and the FCW application which required heavy processing and use of the camera due to its dependence on plate recognition, were affected by the hardware limitations of the smartphones.

# Table of Contents

# List of Figures

# List of Tables

# List of Acronyms

## 0

| | |
|---|---|
| 2G | Second-Generation Wireless Telephone |
| 3GPP | Third Generation Partnership Project |
| 3G | Third-Generation Wireless Telephone |
| 4G | Fourth-Generation Wireless Telephone |
| 5G | Fifth-Generation Wireless Telephone |

## A

| | |
|---|---|
| ACC | Adaptive Cruise Control |
| AGF | Advanced Greedy Forwarding |
| AODV | Ad-hoc On-Demand Distance Vector |
| AOT | Ahead-of-time |
| API | Application Programming Interface |
| APK | Application Package Kit |
| ART | Android Runtime |

## C

| | |
|---|---|
| C2C-CC | Car-to-Car Communication Consortium |
| CALM | Communication Access for Land Mobiles |
| CAN | Controller Area Network |
| CAR | Connectivity Aware Routing |
| CDMA | Code-Division Multiple Access |
| CEN | European Committee for Standardization |
| CVIS | Cooperative Vehicle-Infrastructure Systems |

## D

| | |
|---|---|
| DHCP | Dynamic Host Configuration Protocol |
| DNAT | Destination Network Address Translation |
| DNS | Domain Name System |
| DRiVE | Dynamic Radio for IP Services in Vehicular Environments |
| DSRC | Dedicated Short-Range Communications |
| DSR | Dynamic Source Routing |

## E

| | |
|---|---|
| EDGE | Enhanced Data rates for GSM Evolution |
| EEBL | Emergency Electronic Brake Lights |
| EMS | Emergency Medical Services |
| ESC | Electronic Stability Control |
| ESO | European Standards Organisation |
| ETSI | European Telecommunications Standards Institute |

## F

| | |
|---|---|
| FCW | Forward Collision Warning |
| FHWA | Federal Highway Administration |
| FPS | Frames Per Second |

## G

| | |
|---|---|
| GeOpps | Geographical Opportunistic Routing for Vehicular Networks |
| GLS | Grid Location Service |
| GPCR | Greedy Perimeter Coordinator Routing |
| GPRS | General Packet Radio Service |
| GPSR | Greedy Perimeter Stateless Routing |
| GPS | Global Positioning System |
| GSM | Global System for Mobile Communications |
| GSR | Geographical Source Routing |

## H

| | |
|---|---|
| HAL | Hardware Abstraction Layer |
| HD | High Definition |
| HSDPA | High Speed Downlink Packet Access |
| HTTP | Hypertext Transfer Protocol |

## I

| | |
|---|---|
| IEEE | Institute of Electrical and Electronics Engineers |
| IETF | Internet Engineering Task Force |
| IPC | Inter-Process Communication |
| IPv6 | Internet Protocol version 6 |
| IP | Internet Protocol |
| ISO | International Organization for Standardization |
| ITS | Intelligent Transportation System |

## J

| | |
|---|---|
| JPEG | Joint Photographic Experts Group |

## L

| | |
|---|---|
| LAD | Least Absolute Deviations |
| LBP | Local Binary Patterns |
| LiDAR | Light Detection and Ranging |
| LTE | Long Term Evolution |
| LTS | Least Trimmed Squares |

## M

| | |
|---|---|
| MAC | Medium Access Control |
| MJPEG | Motion JPEG |

| | |
|---|---|
| MSE | Mean Squared Error |
| MSs | Mobile Sensors |

## N

| | |
|---|---|
| NAT | Network Address Translation |
| NEMO BS | Network Mobility Basic Support |
| NHTSA | National Highway Traffic Safety Administration |

## O

| | |
|---|---|
| OBD | On-Board Diagnostics |
| OBU | On-Board Unit |
| OCR | Optical Character Recognition |
| OEM | Original Equipment Manufacturer |
| OSM | OpenStreetMap |
| OS | Operating System |

## P

| | |
|---|---|
| P2P | Peer-to-Peer |
| PGB | Preferred Group Broadcasting |
| PNG | Portable Network Graphics |
| PSNR | Peak Signal-to-Noise Ratio |

## Q

| | |
|---|---|
| QoS | Quality of Service |
| QVGA | Quarter Video Graphics Array |

## R

| | |
|---|---|
| RAM | Random Access Memory |

| | |
|---|---|
| REST | Representational State Transfer |
| RLS | Reactive Location Service |
| RPDB | Routing Policy Database |
| RSA | Road Safety Authority |
| RSUs | Road Side Units |
| RTSP | Real Time Streaming Protocol |
| RTTT | Road Transport and Traffic Telematics |
| RTT | Round-Trip Time |

## S

| | |
|---|---|
| SAPs | Sensor Access Points |
| SAR | Spatial Aware Routing |
| SDK | Software Development Kit |
| SNAT | Source Network Address Translation |
| SSIM | Structural SIMilarity |

## T

| | |
|---|---|
| TBF | Trajectory-Based Forwarding |
| TCP | Transmission Control Protocol |
| TICS | Telephone Interview for Cognitive Status |
| TISPAN | Telecommunication and Internet Converged Services and Protocols for Advanced Networks |
| TIS | Traffic Information System |
| TRR | Terminode Remote Routing |

## U

| | |
|---|---|
| UDP | User Datagram Protocol |
| UMTS | Universal Mobile Telecommunications System |
| USB | Universal Serial Bus |

## V

| | |
|---|---|
| V2I | Vehicle to Infrastructure |

| | |
|---|---|
| V2V | Vehicle to Vehicle |
| V2X | Vehicle to Everything |
| V3 | Vehicle-to-Vehicle Live Video |
| VADD | Vehicle-Assisted Data Delivery |
| VANETs | Vehicular Ad-Hoc Networks |
| VCGs | Voice Chat Groups |
| VGA | Video Graphics Array |
| VITP | Vehicular Information Transfer Protocol |
| VM | Virtual Machine |
| VNs | Vehicular Networks |
| VoIP | Voice over Internet Protocol |
| VSNs | Vehicular Social Networks |

## W

| | |
|---|---|
| WAVE | Wireless Access in Vehicular Environments |
| WHO | World Health Organization |
| WiMAX | Worldwide Interoperability for Microwave Access |
| WLAN | Wireless Local Area Network |

## X

| | |
|---|---|
| xDSL | x-Digital Subscriber Line |

# 1 Introduction

## 1.1. Problem Statement

The number of vehicles worldwide, without taking into consideration off-road vehicles and heavy construction equipment, crossed the one billion mark in 2010 [7]. It is estimated to reach 2 billion by 2020, with the highest predicted annual growth rate of 7 to 8 percent shown by China and India. On the contrary, predictions for the United States show a slower growth rate of less than 1 percent, while the rise in the number of vehicles in Western Europe is expected to remain within 1 to 2 percent.

According to a study [1] conducted by the World Health Organization (WHO) on road safety involving 180 countries, the number of road traffic deaths was estimated at 1.25 million per year. Furthermore, as depicted in figure 1.1, the report on road safety by WHO enlisted road traffic injuries as the major cause of deaths among people aged 15–29 years, in 2012.

Thus, with the ever increasing number of vehicles on-road, accidents on road may be depicted as one of the leading problems worldwide, causing not only physical and economic losses to the victims, but also affecting the people present near the site of the casualty directly or indirectly. Occurrences of hazardous collisions also put a lot of stress on the economy of a country due to the incurred infrastructural damages and treatment costs.

Although vehicle safety standards has received its fair share of attention, resulting in the incorporation of advanced technologies like windshield defrosting

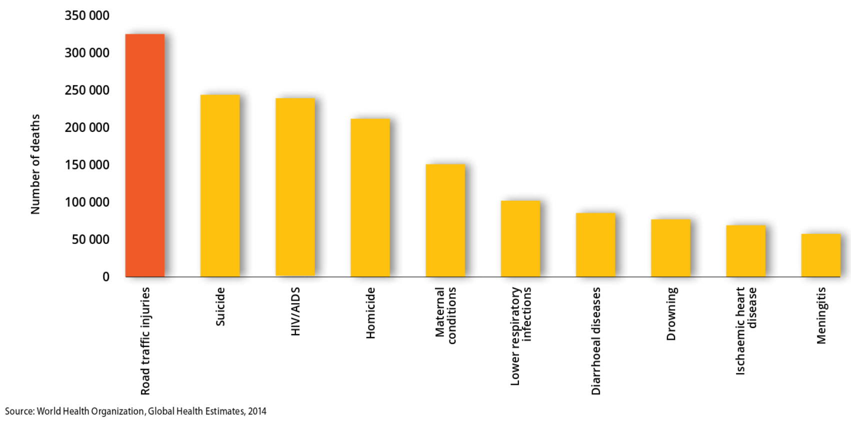


*Figure 1.1: Top ten causes of death among people aged 15–29 years, 2012 [1].*

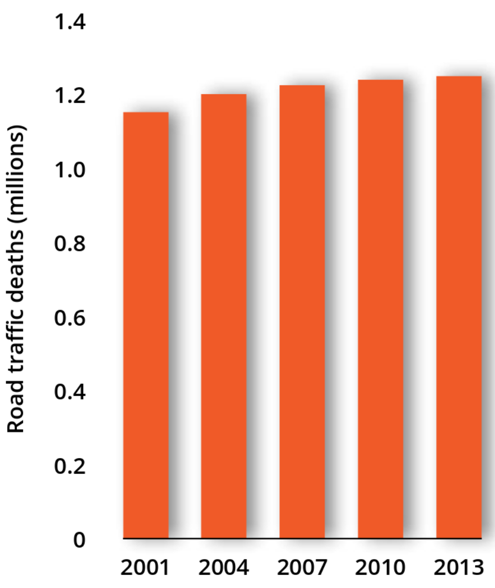


*Figure 1.2: Number of road traffic deaths per year, worldwide [1].*

[8] and defogging, Electronic Stability Control (ESC) [9] and child restraint systems [10], apart from the traditional equipments including seat belts and airbags to improve the safety of the passenger. However, the number of deaths on road due to traffic accidents are on the rise every year, as can be seen from figure 1.2.

ITS [11] is a promising area which may take the advantage of the advances in wireless communication, sensor technologies and computation capabilities aiming to make transport safer, efficient, and more sustainable. VNs [12] is one such communication technology that makes use of different types of networks like cellular, Vehicular Ad-Hoc Networks (VANETs) [13] and other networks, to allow Vehicle to Vehicle (V2V) [14] and Vehicle to Infrastructure (V2I) [15] communications.

Thus, making use of the information obtained from the surroundings, including traffic and road conditions, exchanging sensor information with other vehicles or near-by Road Side Units (RSUs), ITS open a new horizon for traffic safety applications. This thesis is based on intelligent applications that aim to make driving a safer experience for drivers in the context of ITS.

## 1.2. Motivation and Objectives

As a result of the improvements seen in the field of electronics and software, cars too have come across rapid advancement in terms of the assistance they provide to drivers by collecting information from the surroundings and providing important notifications or alerts when required. This helps in avoiding accidents and making life much easier for drivers. Currently, with the help of the efforts of major manufactures and research organisations, the automobile industry is ready to take the next big leap towards autonomous driving, where instead of notifying the drivers, the vehicle itself would react to different situations while in motion. This would eliminate human factors like judgemental errors, slow reactions and poor spacial awareness from the equation, hence eradicating some of most important factors that are responsible for accidents. Although autonomous vehicles are a part of a promising field of research, but they are still in their early stages, with their own set of challenges [16], and it may take some time before we find self-driving cars on the streets.

Slow adoption rates can also be seen affecting various technologies when speaking of intelligent vehicles, the most representative example of which is the inclusion of the IEEE 802.11p standard [17], also known as Wireless Access in Vehicular Environments (WAVE) standard, which allows V2X communication. Thus, delayed incorporation of recent technologies by the automobile industry denies common people the advantages of the different existing ITS applications. The objective of this dissertation is to take advantage of technologies that people make use on a daily basis to provide a platform for the endorsement of different ITS safety applications.

As pointed out, cars that people use on a daily basis remain unable to communicate with one another. So, to facilitate the use of ITS applications that require the exchange of information between vehicles, we will present GRCBox, a low cost connectivity device that allows V2X communication. Making use of the GRCBox for inter-vehicular communication, we designed Messiah, a safety application for smartphones, that merges location and map based information to alert drivers of the presence of important administrative vehicles around them. It also shows on-screen, the possible future route of these vehicles. Another important feature of the Messiah application is that drivers in distress can request other near-by vehicles for help.

The adoption of smartphones, is not only justified by their wide availability and use, but also because they are evolving towards high performance terminals with multi-core microprocessors packed with a wide variety of onboard sensors. Hence, we make use of the computation power and sensors that people carry in their pockets on a daily basis for collecting and processing data. This information, collected from the sensors by the different ITS applications in the context of this dissertation, is later exchanged using the network of vehicles created with the help of GRCBox.

We also provide a detailed explanation of a Forward Collision Alert application that uses image processing techniques to calculate the distance between the two vehicles and warns drivers on getting too close to the vehicle ahead. Drivers, upon being alerted, may choose to brake and maintain safe distance from the vehicle in front. Lastly, we present EYES, an ITS application that is aimed towards providing visual assistance to drivers while overtaking. It makes use of the camera of the smartphones of drivers to record a view of the road ahead, and sends it in real-time to the vehicle behind whenever requested for it, using the vehicular network created using GRCBox. This video is displayed on the screen of the smartphone of the driver in the car behind. Based on this video aid, a decision whether to overtake, can be taken. This application is specially useful when the view of the driver is blocked by the presence of a larger vehicle in front of it.

It can be seen that all the works belonging to this thesis, make use of VNs and smartphones for the implementation of the different ITS applications, making it quite clear that our goal here is to study the integration of smartphones with vehicular networks to develop ITS applications that can reach out to the masses in a short period of time.

## 1.3. Structure of the Thesis

This dissertation is organised in five parts. The contents of each part are described below:

1. Introduction - The first chapter which includes a brief foreword on ITS, its applications, the motivation behind choosing this field of research, and the problems to be addressed in this thesis.

2. Background - Explains some of the most important technologies and standards related to this dissertation, and describes the importance of some of the works that are closely related to our own.

3. Preliminary contributions - Presents the three preliminary works including a connecting device used to establish vehicular communications called GRCBox; a smartphone application that alerts drivers of the presence of important vehicles around it, named Messiah; and lastly, a FCW generation system.

4. EYES: The Video Overtaking Aid - This chapter focuses on an ITS application that aims to provide visual assistance to drivers while overtaking, and describes the results obtained in the experiments with the developed application.

5. Overall Conclusion - Lists the inferences from the experiments described in the previous chapters, and details the possible improvements that can be made to the works presented. This chapter also incorporates the publications related to this thesis.

# 2 Background

## 2.1. Vehicular Networks & ITS

VNs include all those kinds of networks where one or more nodes are vehicles, usually equipped with an On-Board Unit (OBU) for connectivity. An OBU is a small computer acting as a router for forwarding data. Figure 2.1 shows a network of vehicles where the nodes are able to communicate with one another.

Participating nodes in a vehicular network have significantly distinct characteristics and demands from traditional wireless adhoc network nodes that function in an infrastructure-less environment. Vehicles, when compared to a typical mobile computing node, have a much higher power reserve because power can be drawn from a battery which is recharged as the vehicle moves. Also, due to their larger size and plentiful energy supply, vehicles can adhere to larger storage, heavier computing and greater sensorial needs. They can further be equipped with powerful transceivers capable of faster data delivery rates. In spite of these advantages, mobility in VNs make it challenging to maintain communication over a large period of time. However, existing statistics of vehicular motion, for example the typical trend to travel together, or traffic patterns during peak hours, help in maintaining connectivity between vehicular groups.

ITS is a technology, application or platform, which without embodying intelligence as such, aims to improve safety, mobility and efficiency of ground transportation systems, making use of sensing, analysis, control, and communication technologies. ITS includes a wide range of applications that process and share

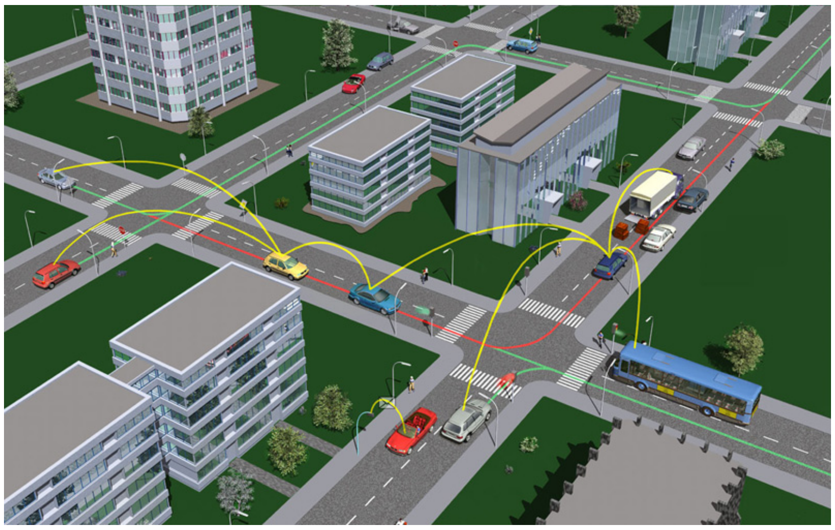


*Figure 2.1: Communication among vehicles [2].*

information, enabling users to be better informed, improve traffic management, minimise environmental impact, and increase the benefits of transportation to commercial users and the public in general. Thus, we see that ITS and VANETs are closely related to one another. In this thesis we will design, develop, and evaluate various ITS safety applications for smartphones, since our goal is to study the effects of integrating smartphones to VNs.

### 2.1.1. Communication in Vehicular Networks

In this section we will discuss some of the technologies that are employed by researchers to make V2V and V2I communications a reality.

#### 2.1.1.1. DSRC/WAVE

Dedicated Short-Range Communications (DSRC) operates in the 5.9 Ghz range, and it is a short to medium range communication technology [18], endorsed by the Standards Committee E17.51. It is a variation of the IEEE 802.11a Medium Access Control (MAC) for the DSRC link, also known as WAVE [19], when considering vehicular network scenarios. DSRC has two modes of operation, namely *adhoc* and *infrastructure* mode, and supports vehicle speeds up to 120 mph, a nominal transmission range of 300 m (up to 1000 m), and a default data rate of 6 Mb/sec (up to 27Mb/sec).

The *adhoc* mode is characterised by distributed multi-hop networking (V2V), while *infrastructure* involves a centralised single-hop network (V2I or vehicle-gateway). It should be noted here that, depending on the deployment scenarios, gateways can be connected to one another or to the Internet, and they can be equipped with computing and storage devices. An example of which is Infostations [20].

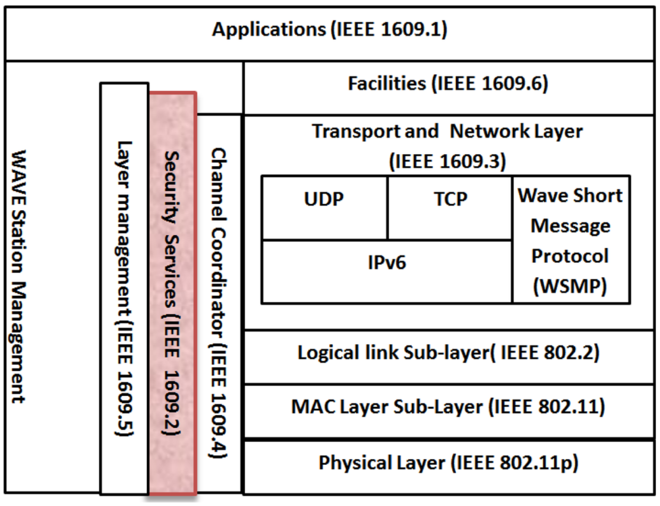

Applications (IEEE 1609.1)
Facilities (IEEE 1609.6)
Transport and Network Layer (IEEE 1609.3)
UDP
TCP
IPv6
Wave Short Message Protocol (WSMP)
Logical link Sub-layer( IEEE 802.2)
MAC Layer Sub-Layer (IEEE 802.11)
Physical Layer (IEEE 802.11p)
WAVE Station Management
Layer management (IEEE 1609.5)
Security Services (IEEE 1609.2)
Channel Coordinator (IEEE 1609.4)

*Figure 2.2: Layered architecture of the WAVE standard [3].*

As shown in figure 2.2, WAVE encompasses two standards, namely the IEEE 802.11p [21] which includes the physical and MAC layer specifications; and the IEEE 1609 standards [22] for the other higher layers for managing resources (IEEE 1609.1), security (IEEE 1609.2), addressing and routing (IEEE 1609.3), coordination of channels (IEEE 1609.4), management of layers (IEEE 1609.5), and finally managing application facilities (IEEE 1609.6).

### 2.1.1.2. Cellular Networks

Cellular systems have seen rapid evolution as a result of the ever increasing demand and popularity of mobile networking. The support for data communication began with Second-Generation Wireless Telephone (2G) Systems such as IS-95 [23] and Global System for Mobile Communications (GSM) [24], which provided a maximum data rate of 9.6 kbps. To provide higher data communication rates, IS-95 based Code-Division Multiple Access (CDMA) [25], General Packet Radio Service (GPRS) and Enhanced Data rates for GSM Evolution (EDGE) are used. The data rates for IS-95 based CDMA is below 141 kbps, while the speed of GPRS and EDGE is under 171 and 384 kbps, respectively. Third-Generation Wireless Telephone (3G) systems support higher data rates, when talking of Universal Mobile Telecommunications System (UMTS) or High Speed Downlink Packet Access (HSDPA), achieving speeds of 144 kbps, 384 kbps and 2 Mbps depending on whether the scenario involves high mobility, low mobility or is stationary. And CDMA 2000 1xEvDO (Rev. A) [26] provides 3 Mb/sec and 1.8 Mb/sec for down and up links, respectively. Currently we are in the Fourth-Generation Wireless Telephone (4G) [27] era, which is defined by the Third Generation Partnership Project (3GPP) and based on a highly flexible Long Term Evolution (LTE) radio access technology. Systems based on the first release of LTE, 3GPP Release 8 finalised in 2008, is capable of providing downlink and uplink peak rates up to 300 and 75 Mbps, respectively [28]. Slowly we are moving towards Fifth-Generation Wireless Telephone (5G) [29] technology, which is expected to be deployed soon.

While we have taken a look at the theoretical data rates of different networks, in practise the average data rate experienced by users is much lower. GSM/EDGE actually provides less than 128 kbps, and 3G under 512 kbps. Qureshi et al. [30] studied the behaviour of 3G services (1xEvDO) in the vehicular environment, and discovered that the average Round-Trip Time (RTT) was quite high, in the order of 600 ms. They also experienced a small number of disconnections of less that 30 sec during the experiments. Another interesting observation from the experiments conducted by Qureshi et al. was that the download throughput ranged from 100 to 420 kbps, while the peak upload throughput was less than 140 kbps. Also, no correlation was found between the velocity of the moving vehicle and the achieved throughput. Nevertheless, location played a dominant role leading to the variation in the throughput achieved.

#### 2.1.1.3. WLAN

In 1996, the IEEE started to create standards for Wireless Local Area Network (WLAN), which can also support broadband wireless services. The first standard was out in 1997, and was revised in 1999. Ubiquitous deployment possibilities make WLAN an attractive method to support broadband wireless services. It has long been used as a means of Internet access in vehicles, known as Wardriving [31].

Among the different standards that exists, 802.11b [32] has the lowest speed with 11 Mbps, while 802.11 a/g [33] provides 54 Mbps and has a transmission range of about 30 meters. Other standards, like 802.11n [34], 802.11ac [35] and 802.11ad [36], provide about 600 [37], over 6000 [38], and nearly 7000 Mbps [39], respectively. The newest standard, 802.11ax [40], to be publicly released sometime in 2019, promises higher speeds of 10000 Mbps [41]. Also, there are two active IEEE projects, 802.11ay [42] and 802.11az [43], which might deliver even higher data rates.

#### 2.1.1.4. WiMAX/802.16e

Worldwide Interoperability for Microwave Access (WiMAX), or 802.16e [44], is an alternative to cable and x-Digital Subscriber Line (xDSL) [45], which aims at the delivery of wireless broadband access over long distances with speeds of under 40 Mbps. This allows fulfilling the gap between 3G and WLAN standards, providing data rate of the order of tens of Mbps, supporting mobility (under 60 km/h) with extensive coverage of about 10 km. Another similar technology, also based on 802.11e standard, is WiBro [46]. WiBro was developed in Korea, allowing a maximum rate of 6 and 1 Mbps for down and up-links per user. It supports services including guaranteed Quality of Service (QoS) for delay sensitive applications, including an intermediate QoS level for delay tolerant applications that requires a minimum guaranteed data rate, and has a range of 1 km for communication purposes.

Han et al. [47] performed some experiments with WiBro in subway scenarios where the maximum velocity reached up to 90 kmph. Some of the key observations from the experiments conducted by Han et al. showed that data speeds of 2 and 5.3 Mbps could be achieved for uplink and downlink, on an average. It was further observed that the average packet delay, which is approximately half the RTT, was less than 100 msec. The delay experienced by most of the packets was below 200 msec, except when handoffs occur, in such cases the measured delay was over 400 msec.

### 2.1.2. Routing in Vehicular Ad-Hoc Networks

VANET routing protocols can be classified into three groups: source routing, geographic routing, and trajectory-based protocols. As the name suggests, in the source routing protocols, the source node decides the route, which normally includes list of points, nodes or subareas that a message should take to reach its destination. Geographic routing, on the other hand, does not directly use geographic routing over the graph made of participating nodes and wireless links; instead, it uses graphs of streets for routing purposes. The third category, called trajectory-based protocols, takes advantage of the prior knowledge of the route of a vehicle for packet delivery. Thus, we are going to summarise some of the principal protocols in each of the groups and their internal sub-classification.

But before moving on to the protocols, let us take a quick look at some of the works that studied the development of location service protocols, which plays a huge role when it comes to routing in vehicular networks. The European CarTALK 2000 [48] project is one such work where vehicles announce their position to others over a limited number of hops, and the update frequency depends on the distance between the source and a particular vehicle. As the distance between the source and other vehicles increases, the update frequency is lowered, thus making it less effective when the destination is far away from the source. Also, to reduce network traffic and help intermediate nodes cache locations of other vehicles, position related information is carried within data packets.

FleetNet [49] is German project where two different location services have been studied, namely the Grid Location Service (GLS) [50] and Reactive Location Service (RLS) [51]. GLS considers the network area as a grid where all participating nodes report their position to servers, which are also vehicles. The servers are located in squares of the grid of increasing size, and the number of servers decreases logarithmically with the distance from the node. When a query is made for the location of a particular vehicle, using geographic routing, the query is forwarded to the known destination with the closest identifier to the destination. This process is repeated hop by hop until a server of the destination is reached, which responds to the query. Experiments show that GLS does not perform well in scenarios involving high mobility and node density [52]. The other approach, which is known as RLS, is based on flooding to find the destination. Even though the flooding algorithm incorporates a neighbour elimination scheme to avoid broadcast storms, it still suffers from scalability issues.

V-Grid [53] is an approach that makes use of two complementary location services, one taking advantage of infrastructures, and the other considering the ad-hoc nature of the vehicular network. Thus, the position of the vehicles can be easily acquired when there is access to infrastructure networks, but it also works when the network is isolated.

#### 2.1.2.1. Source-Routing-Based Protocols

This group of protocols, also referred to as source routing technique, is useful when the intermediate hops are locations instead of a particular node.

- Geographic Source Routing - Geographical Source Routing (GSR) [54] was developed when traditional routing protocols like Dynamic Source Routing (DSR) [55], Ad-hoc On-Demand Distance Vector (AODV) [56] and Greedy Perimeter Stateless Routing (GPSR) [57] failed to perform in VANETs. GSR is based on Dijkstra's algorithm [58] to find the shortest path, and uses the knowledge of the street map of the area where nodes are deployed. The list of crossroads is included in the header of the message and transmitted in a greedy [59] manner to the 1-hop neighbour located closer to the next crossroad. This method of routing suffers when there is low connectivity due to few vehicles present. Also, an on-demand location service protocol is used as a tool by the source to locate the destination. This location service in turn uses global flooding, which limits the scalability of the protocol. In terms of performance, it was observed that GSR achieves results similar to AODV, and it is slightly better than DSR.

- Spatial Aware Routing - The novelty of Spatial Aware Routing (SAR) [60] is the introduction of a strategy to avoid packet losses when a local maximum is reached, even though it uses GSR as the base protocol. Upon detecting the lack of 1-hop neighbours that could forward the packet towards the next intermediate crossroad included in the source routing header, instead of discarding the packet like in GSR, the protocol tries to find an alternate route. If the current node is a local maximum, it recomputes the shortest path from itself to the next point using Dijkstra's algorithm once again, eliminating the edge representing the current street. A packet may be further stored while waiting to find the next hop neighbour, to avoid discarding it. When comparing the performance of SAR with GPSR, it was found that SAR achieves a better packet reception ratio, and it performs better in networks of high density as its overhead is independent of network density, but usually the paths used are much longer in comparison.

- A-STAR - Proposed by Seet et al. [61], it is similar to GSR and Terminode Remote Routing (TRR) [62] where the source using Dijkstra's algorithm over the street map determines the trajectory of the message being sent. The novelty introduced by the authors to this protocol is the use of traffic density of each street as weight assigned to the edges. This means that routes with a high probability of message delivery will be used. It also employs a similar recovery strategy as SAR, being that when using alternate routes, information regarding current and temporary states of streets where routing problems exists is also disseminated. This information can be used by the nodes involved in forwarding data to update the graph, although this knowledge is discarded over time assuming that information can be outdated due to the constant changes in the state of the network.

- Connectivity-Aware Routing - There exist two major problems in most protocols of this category, that is, the use of geographic routing algorithms over a graph of nodes instead of a graph of streets, and secondly, the assumption that nodes on the same street are connected, is not always true. Connectivity Aware Routing (CAR) [63] is a source-routing-based protocol that applies a greedy scheme called Advanced Greedy Forwarding (AGF) [64] to progress through the crossroads, trying to solve these issues. Instead of using the Dijkstra's algorithm, it uses a preliminary flooding phase based on the Preferred Group Broadcasting (PGB) [64], to locate the destination. The response from the destination travels back to the source following the inverse tree created in the flooding phase. While the message is travelling towards the source, intermediate nodes may add their location to the message if they are situated close to crossroads. Thus, the source receives a list of anchor points that may be used as source routing header. A node may detect that it is located near a cross-road with the help of the periodic beacons it receives from its neighbours, where the beacon contains the velocity vector of the sender. Thus, on obtaining beacons from two different 1-hop neighbours whose velocity vectors are not parallel, it can be assumed that the node is near a crossing. Even though CAR outperforms GPSR, it suffers from scalability issues as the protocol is dependent on an initial broadcast.

#### 2.1.2.2. Geographic-Routing-Based Protocols

The idea behind the protocols discussed under this section is the application of geographic routing techniques directly over the map of streets to decide the next direction on reaching a cross road. Note that these protocols do not keep augmenting the header of the messages, as in case of source routing.

- Greedy Perimeter Coordinator Routing - The developers of the Greedy Perimeter Coordinator Routing (GPCR) [65] protocol do not assume the knowledge of the complete street map of the area, neither do they rely on flooding or source routing headers, thus achieving a good delivery ratio with a low control overhead. To be able to apply geographic routing directly over the street graph, GPCR assumes it to be planar with crossroads as vertices, and streets as edges. Coordinators are nodes located at junctions, and responsible for routing the packets in a greedy manner towards the junction closer to the destination, basing the decision on the availability of streets. When routing in perimeter mode, the right-hand rule [66] is applied to determine the next junction. On the contrary, a normal node only forwards packets to the next farthest neighbour along the same street towards the next junction. To determine the role of a node as either normal or coordinator, there are two approaches, the first one where a coordinator node must have two 2-hop neighbours which are within the communication range of one another, but not present in each other's list of neighbours. This approach leads to confusion when vehicles are located on a curve. The second approach involves a statistical method for determining whether a node is located at a junction, to be able to act as a coordinator. Finally, the coordinators advertise their presence to others by making use of beacons which include their role and location information.

- VADD - Presented in [67], the Vehicle-Assisted Data Delivery (VADD) tries to address problems caused by high mobility and low connectivity in vehicular networks. The authors proposed a mathematical formula to estimate the transmission speed of each street segment taking into consideration factors like traffic density, speed limit, and mean vehicle speed. Using this formula, a node located near a crossroad may decide whether to continue carrying it or forward it to another node. Once a decision has been reached so as which street to take, the next challenge is to choose the next relay.

For selecting the next node, different strategies exist, one of them is selecting the node closest to the next intersection. This method can lead to the creation of cycles when the selected node is moving away from the destination; this may avoided up to n nodes by adding the previous n hop information in the header. Another strategy which does not produce cycles, but inefficient in terms of number of hops or end-to-end delay of the paths, involves selecting the node whose movement is aligned to the relative position of the current next intersection selected. The last approach is a multipath one where more than one next relay is chosen. Although it has better delivery ratio and end-to-end delay, the overall network traffic overhead is high.

VADD also uses a store-and-forward approach to compensate for the unstable nature of the vehicular network. Thus, when a node carrying a packet is travelling towards the intersection that the packet has to reach next, it is forwarded using geographical greedy routing if there is a neighbour available, otherwise the current packet carrier continues to carry it. Similarly, if the packet carrier is travelling away from the target intersection, it holds on to the packet until it comes across a vehicle travelling in the opposite direction.

#### 2.1.2.3. Trajectory-Based Protocols

Most vehicles today are equipped with navigation systems that make use of GPS [68] to determine their position on street level maps. These devices sometimes have the capability to connect to the internet on the move for periodic updates of useful information such as weather, location of places like restaurants and gas stations, density of traffic and road conditions. These kinds of information can be useful for routing purposes. For example, prior knowledge of the route that a vehicle is going to follow may come in handy in the process of taking routing decisions. Next, we take a look at some of the most important protocols based on this concept.

- Trajectory-Based Forwarding - Introduced by Niculescu et al. [69], the first protocol to take advantage of predefined trajectory information to guide routing decisions was Trajectory-Based Forwarding (TBF). The idea behind this approach is to determine and include in the message header, a parametric equation for an imaginary trajectory that should be followed to reach the destination. For routing a packet, the next relay neighbour has to be a point on the curve ahead of the last one, and it is randomly chosen among: the neighbour closest to the curve on a straight line, the node whose closest point on the curve results in the greatest advance along the curve, and the neighbour closer to the centroid of the candidate neighbours. Also, choosing a neighbour whose trajectory fits the best with the imaginary curve is a method employed when nodes are not stationary, and it is possible to collect the direction and speed information of the neighbours.

- Opportunistic Geographical Routing - In 2007, Leontiadis et al. [70] proposed an opportunistic forwarding approach for VANETs that assumes that nodes have prior knowledge of their route along the map which they exchange with each other using beacons, and that the position of the destination is already known to the source. Routing in Geographical Opportunistic Routing for Vehicular Networks (GeOpps) is based on a simple forwarding strategy, and it is dependent on the expected trajectory of the neighbours. Thus, packets are forwarded to a neighbour whose trajectory would take them closer to the destination than the current node; this way, the packets move closer and closer to the destination, and eventually are delivered. But since the nodes have the route and speed information of the neighbours, more sophisticated methods of delivery can be used by this protocol where the next hop may be chosen based on the minimum estimated delivery time, which in turn is the sum of the estimated time to reach the nearest point and the estimated time from there to the destination. However, cases might arise where packets fail to reach the destination, due to the fact that sometimes the route is not close enough to the destination to allow direct transmission, or no other neighbour is able to carry the packet closer to the destination.

## 2.2. Vehicular Network Applications

Applications developed for vehicular networks exploit data collected from vehicles, and may make use of infrastructure, aiming to improve safety, better manage traffic, and also provide comfort and entertainment. Here we have categorised applications into three groups: safety-related, emerging, and smartphone-based. Safety-related applications are the ones that alert drivers when there is a high probability of accident occurrence. This category also includes some applications that are autonomous in nature, like adaptive cruise control. In the section emerging applications, we have further sub-classified applications into data source, data consumer, and data producer/consumer applications depending on how they manage data. Lastly, since this thesis is focused on ITS safety applications for smartphones, we describe some existing applications in the literature that were aimed at smartphones.

### 2.2.1. Safety-Related Applications

Many safety applications that we come across do not involve communication between vehicles. In such a case, data is not shared with nearby vehicles, and hence, their capabilities are limited. An example of such an application is a parking sensor, which is a type of proximity sensor that alerts drivers of nearby obstacles. According to some authors [4], the incorporation of communication would enable the use of a wide variety of applications, which can be classified as follows:

1. Vehicle diagnostics and maintenance.
2. Intersection collision avoidance.
3. Public safety.
4. Sign extension.
5. Vehicle cooperation.

Next, we will take a detailed look at each of the above mentioned categories.

#### 2.2.1.1. Vehicle diagnostics and maintenance

The applications that fall under this category are *Safety recall notice* and *Just-in-time repair notification*. These applications provide vehicle owners with reminders related to safety defects and maintenance schedules of their automobiles, and generally involve V2I communication. In case of the United States, the National Highway Traffic Safety Administration (NHTSA) [71] is the agency that is in charge of issuing recall information about vehicles. This information is based on violation of the federal motor vehicle safety standards or safety-related defects associated with vehicle equipment, seats, tires, and so on. *Safety recall notice*, as the name suggests, relies on V2I communication to caution drivers when a recall is issued for their automobile. In case of *Just-in-time repair notification*, the in-vehicle system initiates the communication with the infrastructure upon detecting problems related to the vehicle. The message is forwarded to the Original Equipment Manufacturer (OEM) technical support centres, where the issue is analysed and instructions are sent to drivers, suggesting the best way to deal with such a problem.

#### 2.2.1.2. Intersection collision avoidance

This type of safety application also involves V2I communication. Here, sensors are used at intersections to collect information about the activities of nearby vehicles. The data collected is processed and analysed for possible accidents; upon the detection of unsafe conditions, a warning message is sent to vehicles present in the area.

Many situations may cause the generation of such a warning message:

- Traffic signal violation - An analysis is made to detect whether the vehicle is about to run the traffic signal. A message is sent to such a vehicle depending on its speed, position, status of the traffic signal, schedule of the signal, and other parameters like road and weather conditions. A more detailed description of the application can be found in [72].
- Turn assistant - On a road with right hand traffic, taking a left turn is more difficult, and vice versa. Thus, with the help of special sensors, or DSRC [73] radios at signalised intersections, information about vehicles coming from the opposite direction is gathered. Later, this information is passed on to vehicles with the left turn signal on.
- Blind Merge Warning - This system aims to help drivers to avoid accidents when approaching a merge point. If the risk of a merge is high, and there is a possibility of collision, all vehicles that might be involved or affected are warned.
- Pedestrian Crossing - Sensors located on sidewalks that are able to detect pedestrian movement are used by this system. Also, information from intersections equipped with “walk” buttons used by pedestrians to request a green light to cross the street, can be put to use to alert drivers when a pedestrian is attempting to cross the road.

#### 2.2.1.3. Public safety

Public safety applications are designed to assist emergency teams to provide better services by minimising the travel time to the site of an accident, and also aims to help drivers when in need of aid. Some of the popular public safety applications are listed below, and they might make use of either V2V or V2I, or both types of communications:

- SOS services - Uses both V2V and V2I communications for sending messages. This system is basically intended to call for help automatically when the driver is in a life-threatening situation. Vehicles involved in accidents autonomously detect the occurrence of such events, and send out SOS messages. Sometimes this operation may be initiated by drivers manually when no damage has been done to the vehicle, but still assistance is required. If the vehicle is within the range of the infrastructure, messages are sent directly to ask for help, otherwise nearby vehicles are contacted and the message is eventually relayed to the infrastructure.

- Post-crash warning - This type of systems are designed to avoid secondary collisions when an accident has already taken place, or when a vehicle is immobilised for some other reason. The vehicle that is not in motion due to technical problems or accident, warns other nearby vehicles about its situation. Systems of this kind come in handy when visibility is poor. Since both V2V and V2I are taken advantage of in this case, warning messages containing the vehicle's location, heading, and status are shared with the nearby infrastructure, as well as with other vehicles. When other vehicles receive a post-crash warning message, their location and direction will determine the importance of the message based on which the driver of the vehicle is warned.

- Approaching emergency vehicle - Lives may be saved if the response time of emergency vehicles can be reduced. The approaching emergency vehicle warning system is designed to help reduce the travel time of paramedics, fire brigades and police personnel by having the roads clear. The emergency vehicle sends messages to nearby vehicles that it comes across, notifying them to leave passage. The message may contain the location of the emergency vehicle, lane information, speed, and intended route. Here only V2V communication is used.

- Signal preemption - The Federal Highway Administration (FHWA) [74] defines preemption as interruption of normal signal operations to transfer right of way to the direction of an approaching emergency vehicle, although a green indication is not always guaranteed immediately after preemption is requested. This system works by having the infrastructure at each intersection collecting data from messages that are sent by emergency vehicles.

#### 2.2.1.4. Sign extension

The objective of applications under this category is to keep the drivers informed about different road conditions and other signs that we commonly find while driving. The importance of these applications lies in the fact that, sometimes, drivers tend to miss out road signs due to distractions, while signs may also be missing at times. These applications notify drivers of all types of signs on the road, as well as structures such as bridges present in the area.

Examples include alerting drivers while moving through specific regions like zones under construction, school areas, near hospitals, or animal crossing zones. Typically, RSUs are located at sites where the drivers need to be warned. Similarly, a curve speed warning system, with help of RSUs and V2I communication, informs vehicles of approaching curves, and other relevant information related to its location, curve speed limit, curvature, and road surface conditions. On receipt of such a message, the vehicle processes the data received, and if the driver is driving over the speed limit, an alert is generated. Another example of sign extension is a wrong-way driving warning system, which alerts drivers when the vehicle is driven against the direction of traffic. Sometimes areas with low clearance pose a threat to both driver and passengers within a vehicle. These applications inform drivers about the clearance height, when a vehicle approaches zones with low clearance like parking structures, tunnels or areas with low bridges. The in-vehicle system receives these messages, and if the clearance height is insufficient, the driver will be notified.

#### 2.2.1.5. Vehicle cooperation

Cooperation among vehicles in the form of information exchange using short-range communication between a host vehicle and other nearby vehicles can lead to the development of different applications:

- Visibility Enhancer - Solutions designed to enhance poor visibility situations, such as extreme weather conditions including heavy rain, fog, snow storms, and so on. They may combine imaging systems (cameras, sensor arrays, special optics), special illumination systems (LEDs and pulsed lasers), image enhancement through image and data processing techniques, and in some cases, augmented reality displays for this purpose. Through the exchange of data collected among vehicles in a given area, an enhanced view can be presented to the driver.

- Road Condition Warning - This solution informs the driver and other vehicles around about the condition of the road; this is based on the use of special sensors that collect data about road conditions and vehicle stability. The collected data is processed by the system within the vehicles, and the driver is presented with the results obtained. The result, for instance, could be advising the driver on the analysed maximum safe speed limit, which can also be shared with near-by vehicles.

- Emergency Electronic Brake Lights (EEBL) - The main idea behind this application is to alert drivers when a preceding vehicle performs severe braking. Developed in March 2006, it is considered the first application that used V2V communication. It is a OEM funded project by BMW, Daimler Chrysler, Ford, GM, Nissan, and Toyota. Usually, real tail lights are used to indicate the drivers behind that the vehicle ahead is decelerating. However, bad weather conditions can lead to poor visibility, and drivers in that case would not be able to see the rear tail lights until they are too close; this problem led to the development of EEBL [72]. Figure 2.3 shows a sce-

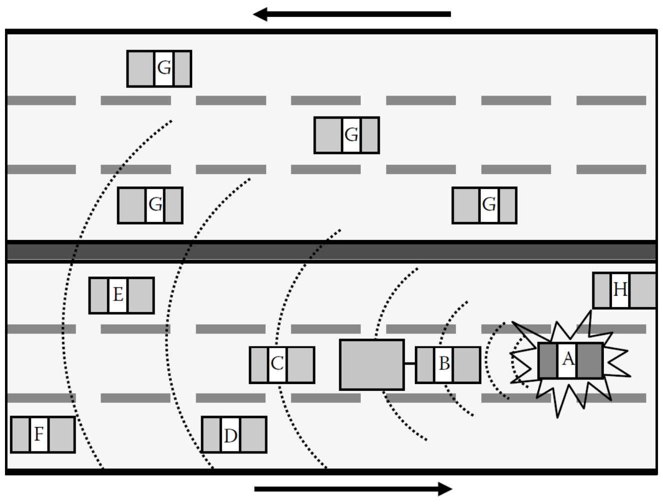


*Figure 2.3: Scenario where EEBL might come in handy [4].*

nario where vehicle A is braking, and this information is transmitted to the vehicles following it.

- Lane Change Warning - The system, in this case, analyses all relevant information like speed, direction, and position of surrounding vehicles when the driver signals for a lane change, to determine if the gap between vehicles is safe enough to make the lane change. If the gap is not sufficient, the system will warn the driver about the risk of the lane change, thus helping to avoid a crash [72].

- Cooperative Adaptive Cruise Control - An improvement over Adaptive Cruise Control (ACC), makes driving safer by controlling the speed of the vehicle based on the velocity of the other vehicles around. Messages exchanged by the application using V2V communication may include information like the position, velocity, acceleration, heading, and yaw-rate. V2I communication may be maintained to further improve performance by gathering information regarding low speed limit zones to maintain speeds below the advisable limit.

- Pre-crash Sensing - Also described in [72], it makes use of sensors present in the vehicles not only to predict the occurrence of an accident, but also to estimate the probable harshness of the accident that is about to happen. Along with information from sensors, the system can take advantage of V2V communications to obtain further information from other vehicles that might get involved in such an accident. Systems such as this can be used to take actions or precautions even before the accident has occurred to reduce damages that could have been caused without it. Examples include the tightening of seat belts before impact.

### 2.2.2. Emerging Vehicular Applications

The prime motivation behind applications using V2V or V2I communication has been safe navigation. However, vehicular networks provide a promising platform for a much broader range of applications given that there is significant demand for more reliability, safety and entertainment in automobiles. This has led to the development of many emerging applications spanning from office-on-wheels, to crime investigation, civic defence, or even mobile gaming.

Some of these applications require internet access for functions like email access or file download while in motion, while, others involve discovering local services while in a particular neighbourhood. Examples include accessing offers from different locals, restaurant menus, and movie schedules from theatres. A couple of other applications may require interaction among vehicles for interactive gaming. To be able to provide all the advanced functionalities to users, the applications perform operations such as maintenance of distributed indices, temporary storage of shareable content, and epidemic distribution of information. Hence, applications in this section, will be classified according to the role of the vehicle in managing data, that is, whether it is acting as a data source, consumer, both, or intermediary.

Vehicular sensor networks is a term used to describe a network where cars are used as a data source to monitor urban environments, and for mobile data gathering [75, 76, 77, 78]. Each vehicle is used to sense data, process the sensed data, and forward this data or related notifications to other vehicles. An example would be to use cameras to capture images of accidents that have occurred, identify the plates of the vehicles involved, and send this information along with the location of the incident to local authorities. Applications of these types require a fast communication rate, persistent and reliable storage of data for later retrieval, and sophisticated networking protocols to efficiently locate and retrieve data of interest.

The next category comprises of applications that are consumers of content, supporting high-fidelity data retrieval and playback. While on a trip, each vehicle is an audience to large quantities of data including locality aware or map-based information, and different types of content for entertainment [79, 80, 81], requiring high throughput network connectivity and fast access to data.

In the third group of compelling applications, vehicles act as both producers and consumers of content. Emergency neighbour alerts (e.g. malfunctioning of brakes), traffic congestion monitoring, and accident reporting are some examples [82, 83, 84] of this category of applications. Finally, it is to be noted here that the above mentioned applications depend on intermediary nodes for temporary storage or caching of data and forwarding information, thus requiring reliable storage, efficient routing, and accurate location information.

#### 2.2.2.1. Data Source

Vehicular Networks are being increasingly put into use for urban monitoring using sensors, and for sharing or retrieval of data. Here we will concentrate on applications that aim to sense and collect data for later use. MobEyes [75, 76] is one such application that falls under the category of data source. This application acts as a middleware, which proactively monitors events on road, locally stores sensor data, and later process the data before forwarding it to the relevant destination. MobEyes may be used to provide aid to law enforcement by helping locate target vehicles, also collecting their trajectory information by making use of plate recognition in the process. Applications of this type requires the collection, storage and retrieval of massive amounts of data since filtering is usually impossible due to the fact that it is unknown beforehand which part of it would be relevant for future investigation. Thus, to reduce network usage, metadata is generated by sensing nodes from the sensed data, which is disseminated over the network so that vehicles interested in the actual data can query for it.

Another sensing platform known as CarTel [77] is based on an intermittently connected database and a web portal where users submit their queries. The database connects to VNs, and it is responsible for dispatching queries to vehicles when they are in the proximity of open access points. Pothole Patrol ($P^2$) [78] is a system to access the road surface condition, and it was proposed by Eriksson et al. It uses data gathered opportunistically by analysing vibrations and combining the collected information with GPS data, to map the points where road conditions are below acceptable conditions. Similarly, Yoon et al. [85], using GPS traces collected from vehicles, proposed a method to access traffic conditions.

VNs can also be coupled with sensor networks forming an opportunistic mobile sensing platform, leading to the development of a wealth of applications. An example is ZebraNet [86], which is used for wildlife tracking. ZebraNet was put to use by the Mpala Research Center in Kenya by equipping zebras with collars housing biometric sensors, GPS, and a wireless transceiver. The collars, equipped with sensors, collect data which is opportunistically exchanged with each other; this way, the data is routed to the base station which may be located in a ranger's truck. Urbanet [87] combines different types of networks to create rich, open sensing environments where people, municipalities, and community organisations can share their resources to give mobile users real-time access to sensed data. Urbanet can be used to warn drivers of various incidents on road like traffic congestion, weather conditions, help look for vacant parking spots and suggest routes. Similarly, Eisenman et al. [88] proposed MetroSense, a three-tier architecture consisting of Mobile Sensors (MSs), stationary Sensor Access Points (SAPs), and online servers for storing or processing of sensed data. The MSs move around in the field, delegating tasks and muling [89] data to SAPs, which in turn acts as gateways to internet where the data servers are located.

#### 2.2.2.2. Data Consumer

Content distribution to nodes in VNs may be required for different reasons like entertainment, other types of application data, and updates/patches for already installed software. SPAWN [79], developed by Nandan el al., is an example of a data consumer application which is a BitTorrent-like [90] file swarming protocol for VANETs. The shared file is given a unique id, divided into pieces, and uploaded to a server connected to the internet. Unique sequence numbers are also generated for each piece of the original file. Users passing by access points connected to the internet, download these pieces and later cooperatively exchange the missing pieces. Similarly, Lee et al. [91] proposed CarTorrent, which uses a simpler form of gossiping for peer discovery since a centralised server cannot be used as in traditional BitTorrent. CarTorrent uses a type of k-hop limited probabilistic gossiping employing a crosslayer approach where route discovery of underlying on-demand protocols plays an important role, and for a piece or peer selection, closest-rarest first approach is adopted. Also, BitTorrent protocols suffer from a coupon collection problem, since acquiring a new piece takes progressively longer as a node collects more pieces. To mitigate this problem, CodeTorrent [92], a solution based on network coding for content distribution was suggested.

Most internet-based business models depend heavily on advertisements for revenue. Thus, vehicles with wireless communication capabilities become a lucrative target for this kind of companies. AdTorrent [80] by Nandan et al. is an extension of physical billboards, which by making use of digital billboards or Ad Stations, delivers location-sensitive commercial advertisements to vehicles. Business owners located close to the billboards can subscribe to digital billboards, with options to disseminate text or multimedia-based advertisements. Examples include trailers of movies currently playing in a theatre nearby, or virtual tours of hotels within a given radius. AdTorrent also lets users receive advertisements according to their interest, putting into use a location-aware distributed mechanism to search, rank, and deliver relevant advertisements.

#### 2.2.2.3. Data Producer/Consumer

An example of applications that are both producers and consumers of content is the Vehicle-to-Vehicle Live Video (V3) Streaming Architecture [84], which allows location-aware video streaming from a remote region of interest. V3 is illustrated in figure 2.4, and it comprises of a video triggering and transmission subsystem, requiring vehicles to be equipped with on-board computing capabili-

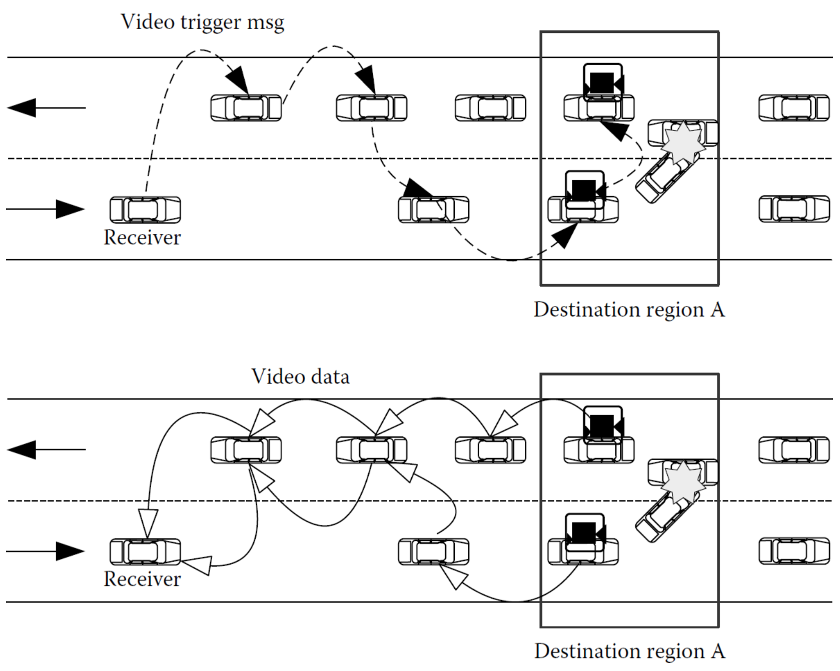


*Figure 2.4: Working of the V3 architecture [5].*

ties, wireless communication devices, GPS, and some of the vehicles need to have cameras with sufficient space for recording, storage or buffering videos. The video triggering subsystem is responsible for forwarding video request messages to the destination region, while the video transmission subsystem is responsible for sending video data back to the receiver. Park et al. [93] proposed an emergency video streaming protocol based on network coding, which pushes urgent video streams regarding situations such as accidents, natural disasters, or other abnormal situations to warn drivers. To provide reliable and robust video streaming, [93] uses random linear network coding. On the other hand, Qureshi el al. proposed Tavarua [30], a 3G-based real-time multimedia communication subsystem aimed at mobile telemedicine applications. Tavarua has three main components, namely Tribe, responsible for low level connection with network interfaces; Horde, which provides services like congestion control and network striping; and lastly, a video services subsystem.

Communication among vehicles can also be used for creating a virtual marketplace where users may sell or buy stuff from others at close proximity. This idea was presented by Lee et al. in their work entitled FleaNet [83]. It does not require any infrastructure support to function, making use of the adhoc grid for communication to find common interests, potentially leading to transactions. Vehicles, as well as static roadside Advertisement Stations (or Adstations), generate and propagate queries, and a driver who received the advertisement could place an order. Vehicular networks are also useful for various on-demand location-aware services such as roadside service directories (e.g., location/menu of a local restaurant), and alerting drivers of different traffic conditions like congestion, traffic flow rates and accidents. This is the function of Vehicular Information Transfer Protocol (VITP) [94], which is a stateless communication protocol that specifies the syntax and semantics of messages carrying location-sensitive queries, or the replies between the nodes. VITP also allows nodes to aggregate information and report the summarised result to the requester.

Exchanging traffic-related information is most common use of VNs. Rybicki et al. [95] designed a distributed Traffic Information System (TIS) that allows vehicles to exchange information related to current traffic status. The developers suggested the use of infrastructure-based mobile wireless access methods such as 3G, GPRS, and WiMAX for better performance since VANETs are still not widely used. Another related application is RoadSpeak [96] presented by Smaldon et al. RoadSpeak is a framework for creating virtual mobile communities, also know as Vehicular Social Networks (VSNs), and it allows commuters to communicate with one another using audio messages on different Voice Chat Groups (VCGs). For this purpose, centralised servers are used, and clients connect to the infrastructure that allows connections to the Internet for downloading the required software, and for the account creation purposes. The creation of the account is important for the generation of the asymmetric key pair, and registration of the public key with the server. Now, the user can choose to join a group and send a request to the group admission manager, who validates the request and sends the group key for future communications. TrafficView [82] is a framework that gathers and disseminates information about the vehicles on road using flooding techniques, providing real-time road-traffic information such as location and speed of vehicles to drivers, thus assisting drivers in situations such as foggy weather, or helping them to find an optimal route during a trip. Since TrafficView uses flooding, data aggregation and fusion based on distance from the source is used to alleviate broadcast storms. Gradinescu et al. [97] showed that adaptive traffic lights based on short-range communication among vehicles and wireless controllers installed in an intersection that determines the optimum values for the traffic light phases, can help to reduce vehicle waiting time, fuel usage and emissions. The authors also argued that this architecture based on communication has more benefits compared to sensors or camera-based adaptive systems. Another application that takes advantage of VANETs is EZCab [98] by Zhou et al., a real-life ubiquitous computing application for discovering and booking cabs in cities using V2V communications. The EZCab application, which works in a decentralised fashion, is easy to deploy, cost effective, and very scalable. The client using the EZCab application sends a BookSM packet to find cabs nearby using flooding or probabilistic forwarding, to which available cabs reply using ReportSM packets. On receiving replies from the cabs, the client may choose one and make a reservation using the ConfirmSM message. Similarly, Huang et al. [99] investigated the financial and technical feasibility of a taxi dispatching application using a realistic mobility and propagation model drawn from real world data.

Lastly, wireless and mobile online gaming is one of the emerging application areas that is growing in importance as wireless networking technology becomes ubiquitous in mobile platforms. At the same time, vehicular entertainment systems are gaining popularity, as a result of which millions of TV/DVD systems are sold every year. Thus, it is only natural that the convergence of VANETs and entertainment will bring online passenger gaming to reality taking advantage of wireless P2P communications. Although internet connectivity through infrastructure or 3G services may also be used for running this application, still infrastructures cannot guarantee the desired quality of service, and 3G connectivity incurs an usage fee. Communication of game data requires high bandwidth and low latency, and is the major challenge of gaming in VANETs [100].

### 2.2.3. Smartphone-based Applications

Both academia and industry have shown very keen interest in ITS solutions that rely on mobile devices, resulting in many innovative applications. We are going to describe some of these interesting works that are closely related to our own, specially concentrating on solutions that are based on smartphones.

One of the first works involving smartphones was published in 2009 by Whipple et al. [101], who developed a safety application that collected the location and speed information using the GPS of the mobile devices; then, using the Google Maps API, they looked up for nearby schools, and alerted the drivers if they drove at a high speed near these school areas.

Two years later, an Android/OSGi-based vehicular network management system was designed by Chen et al. [102]. In the same year, Yang et al. [103] proposed an application that uses the location, moving direction and velocity of the vehicles obtained using the onboard GPS. This information is periodically exchanged between vehicles, and warnings are issued if the probability of collision is high. An Android-based application that detects accidents through the On Board Diagnostics (OBD-II) [104] interface to inform predefined contacts was shown in the work of Zaldivar et al. [105]. CarbonRecorder [106] is another application that can be used to detect the daily carbon emission of vehicles.

The year 2012 saw a lot of development in the field of ITS applications, and one of the applications developed during this period that is worth mentioning is DriveAssist [107] by Diewald et al. DriveAssist provides users with an overview of nearby traffic, triggering warning messages for certain traffic incidents. SMaRT-CaR [108] is a platform also developed during the same period that integrates smartphones and provides support to traffic management applications. Another application that makes use of the OBD-II standard to extract safety and environment-related information was proposed by Wideberg et al. [109]. The See-Through System [110], by Gomes et al., aims at improving the visibility of the drivers using augmented reality components. The developers of SignalGuru [111] predicts the schedule of traffic signals by leveraging collaborative sensing on windshield-mount smartphones.

Later, in 2013, the CarSafe App [112] was introduced by You et al., which is another driver safety app for Android phones that alerts drivers when detecting dangerous driving conditions and driver behaviour. It makes use of computer vision and machine learning algorithms on the images from front and back cameras of smartphones to monitor and detect whether the driver is tired or distracted, while simultaneously keeping track of the road conditions. Another interesting application was proposed by Meseguer et al. [113], which incorporates data mining techniques and neural networks, to analyse and generate a classification of driving styles using the data from the OBD-II interface, assisting drivers to correct the bad habits in their driving behaviour, and also providing tips to improve fuel efficiency.

Other applications available online for download are Waze [114], Torque [115], and iOnRoad [116]. Waze is a well known community-based traffic and navigation application where users share important traffic information on the go. Torque is an Android application that uses the data from the OBD-II connector to monitor different parameters in real-time. The application iOnRoad, on the other hand, provides driving assistance functions including augmented driving, collision warning and, “black-box” like video recording.

Encouraged by the findings from the aforementioned works, we decided to develop solutions that rely on smartphones to achieve rapid acceptance, studying the integration of smartphones with vehicular networks.

# 3 Preliminary Contributions

## 3.1. V2X Communication in Europe

Intervehicular communication was made possible with the recent development and widespread adoption of wireless communication technologies, thus making many proposed ITS applications a reality. The organisation responsible for standards related to the field of the telecommunication industry in Europe is the European Telecommunications Standards Institute (ETSI) [117]. It is a part of the European Standards Organisation (ESO), and well known for its GSM, and Telecommunication and Internet Converged Services and Protocols for Advanced Networks (TISPAN) standards. ETSI is a non-profit entity comprising of members pertaining to academia, private companies, research and public organisations. The working groups that are a part of its technical committee for ITS are focused on application requirements, architecture and cross-layer issues, transport and network, media and its related issues, and security. Another important non-profit organisation is the European Committee for Standardization (CEN), which established the technical committee TC 278 [118] in 1991 for Road Transport and Traffic Telematics (RTTT). The CEN/TC 278 is organised into the following work groups: electronic fee collection, freight and fleet management, public transport, traffic and traveller information, traffic control, geographic data files, road databases, DSRC, human-machine interface, automatic vehicle and equipment identification, architecture, recovery of stolen vehicles, and eSafety.

Similarly, the International Organization for Standardization (ISO) TC 204

[119] which covers basically system and infrastructural aspects of various types of transport, started functioning in 1993. The ISO TC 204 working group that is focused on wide area communications, protocols and interfaces, is responsible for defining a communication system with support for multiple wireless technologies like GSM, 3G, infrared, DSRC, 5.9 GHz ITS, etc; use of plug and play like interfaces; dynamic selection of the optimal interface for use during a session; support for advanced protocols like IPv6 [120] and other extensions for mobility. This framework for a communication system has been named Communication Access for Land Mobiles (CALM). Other working groups that are a part of ISO TC 204 are dedicated to architecture; database; automatic vehicle and equipment identification; guidance and navigation; fee and toll collection; fleet management and commercial freight; public transport and emergency; information, management and control; traveller information systems; warning and control; DSRC for Telephone Interview for Cognitive Status (TICS) applications; and nomadic devices.

Formed in 2002 for creating an open industry standard aiming to achieve interoperability and harmony among the existing standards of V2X communication specially focusing on road safety and traffic efficiency, a consortium of vehicle manufacturers with the help of equipment suppliers, research organisations and other partners in Europe formed the Car-to-Car Communication Consortium (C2C-CC) [2]. C2C-CC consists of six working groups: the first is dedicated to investigating the technical issues of V2X at physical and MAC layers, the networking group focuses on algorithms and protocols for data networking and transport, the third group defines the architecture of the protocols and the system for V2X communication, the application working group specifies the necessary protocols and application requirements as suggested by its name, while the security working group studies the different security issues related to key management and routing protocols, finally the last group is devoted to standardising issues like frequency allocation for communication, and it is responsible for interacting with various standardising bodies.

An open and pragmatic organisation that developed a bulk of the protocols on which Internet users depend, is the Internet Engineering Task Force (IETF). The work of the IETF, which is strongly based on the publication of a draft specification, its review, testing and later re-publication, has contributed strongly towards achieving interoperability. A work that is worth mentioning in this context, and which was a result of the efforts by this group, is the Network Mobility Basic Support (NEMO BS) protocol. This protocol is an extension of Internet Protocol version 6 (IPv6) [121] for an entire moving network, and it has been used in many automotive research projects like InternetCar [122, 123] developed in Japan; the OverDRiVE [124], the follow up of an EU project called Dynamic Radio for IP Services in Vehicular Environments (DRiVE) [125]; and Cooperative Vehicle-Infrastructure Systems (CVIS) [126]. NEMO BS consist of an entity called the mobile router, which is responsible for handling the mobility related issues by updating a binding at a fixed node in the infrastructure network, sometimes known as a home agent. This home agent assures the continuation of a session and data reachability by forwarding packets from the Internet to the current location of the destination.

Thus we see the active participation of the different standardisation bodies and the result of their investigation towards making vehicular communication a reality. In spite of all the hard work, vehicles that end users tend to operate on a daily basis still lacks the capability to communicate among themselves. Hence, people are unable to take advantage of the wide variety of safety and entertainment related applications that VNs support. Aiming to resolve this issue, and to develop safety-related ITS application, we developed the GRCBox, a low cost connectivity device for vehicles that allows plug-and-play installation of wireless interfaces, thus supporting both V2V as well as V2I-based communication.

### 3.1.1. Contribution I: GRCBox

Advanced features in vehicles are normally first introduced in the high end models, thus the same is expected to occur with networking capabilities based on the IEEE 802.11p standard. This will slow down the deployment of ITS applications, as high end vehicles are not so common due to heavy initial and later maintenance costs. Also, the renovation rate of vehicles is quite sluggish as people generally tend to make use of their vehicles as long as possible until a change is absolutely necessary. In such a case, On-Board Units (OBUs) [127] are an option, but these devices are not designed to be updated, hence rendering them obsolete after a few years.

Recently, there has been a lot of interest to integrate smartphones and OBUs owing to the importance, popularity and versatility of these mobile gadgets. Examples of such projects are Mirrorlink [128], Android Auto [129] by Google [130] and CarPlay [131] for iOS devices. Most user actions supported by these projects are voice activated, thus minimising distractions while driving. However, none of these solutions take advantage of the communication capabilities of the smartphones forming a Peer-to-Peer (P2P) network due to their software limitation, and closed design which does not allow adding new networking interfaces. This is one of the biggest reasons behind the limited use of smartphones in ITS. Here, we will present GRCBox, a cost effective solution that aims to help users enjoy sophisticated ITS solutions by making the use of smartphones, thus enabling us to study the effect of integrating smartphones to vehicular networks. GRCBox, being a hardware module that houses multiple interfaces for communications including cellular, wifi, and adhoc, provides many networks for applications to choose from.

#### 3.1.1.1. Architecture

The GRCBox architecture consists of a hardware and software module. The physical device is to be placed within the vehicle, and mobile devices of users in the car can connect to it. Also, since the GRCBox has multiple interfaces, it lets applications decide which interface is to be used for the sending and receiving of data by making use of the software library based on a remote API.

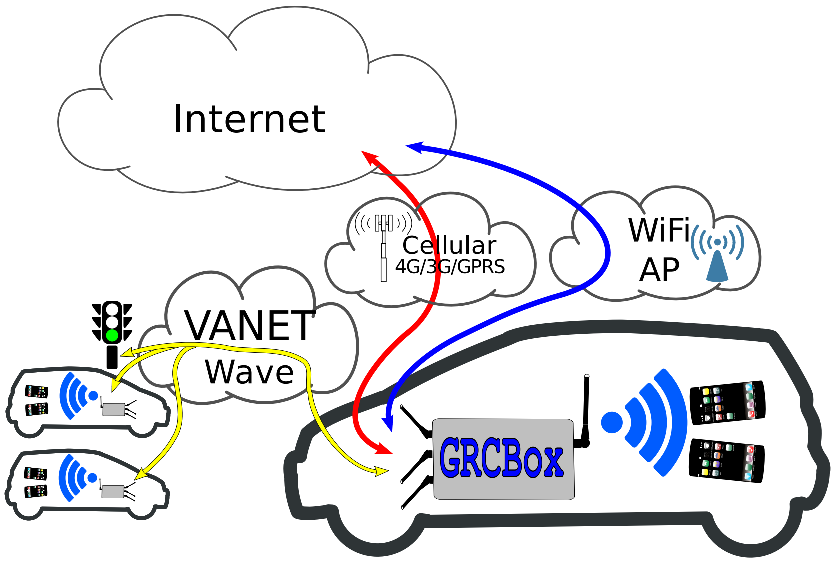


*Figure 3.1: The idea behind the GRCBox.*

Figure 3.1 explains the use of the GRCBox. Here, in this example, users may prefer a stable connection and opt for 3G or LTE services, or wifi, which can be used to connect to open access points for high throughput data transfer; also, there exists the option of adhoc communications that is independent of infrastructure support, providing a direct link between vehicles.

**Components of the Hardware Module** As we can see in figure 3.2, the GR-CBox hardware module, which is allocated inside the vehicle, consists of an optional battery power source since energy can be drawn from the vehicle itself, a controller for processing tasks, an Universal Serial Bus (USB) hub for connecting the network interfaces, and at least two wireless network interfaces. At least two separate network interfaces should be present, as smartphones are able to only connect to wireless access points without being tampered with or rooted. Thus, one of the interfaces (inner interface) is configured to work as an access point so that mobile devices are able to connect to the hardware module, and the other interface

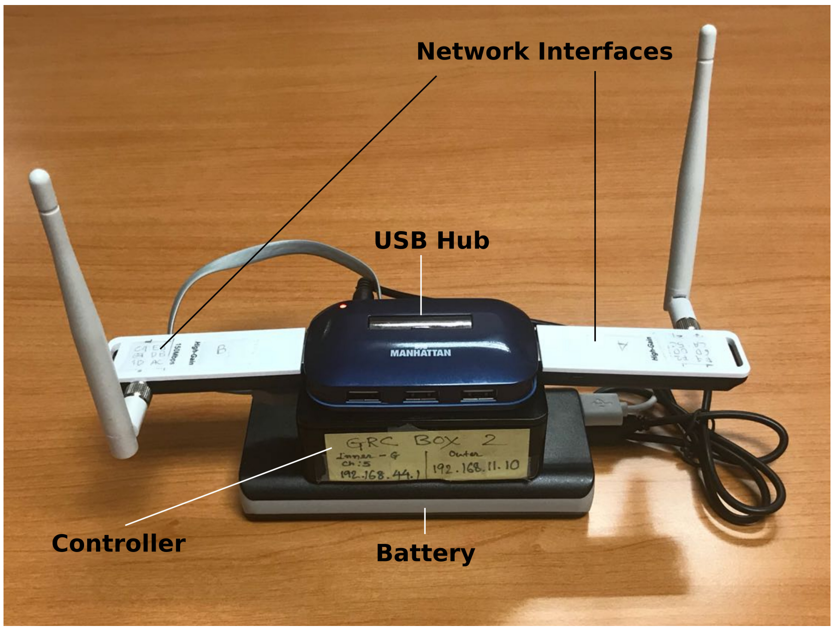


*Figure 3.2: The GRCBox.*

(outer interface) is the one used to forward or receive application data to and from the outside world.

The software support that comes with the hardware is responsible for delivering discovery services, routing packets and performing header modifications, act as multicast and broadcast proxy, monitoring the network interfaces, and fulfilling the core services of the GRCBox. Let us take a look at each of these functionalities in detail:

- Discovery service - Allows clients to detect the presence of the GRCBox and connect to it without making it obligatory for the clients to have any previous knowledge of the GRCBox and the features it supports.
- Routing and header modification - This service receives packets using the inner interface of the GRCBox that the user application or device connects to, and forwards the data received to the relevant outer interface as per the connection rules defined by the application itself. For routing purposes, packet headers are also modified whenever required. The same process takes place when data is received by one of the outer interfaces, and must be forwarded to the inner interface.
- Multicast and broadcast proxy - The module responsible for routing is only able to work with unicast packets; hence, a service that can handle both

broadcast and multicast packets has been defined. This service extends the broadcast domain of the network formed by the devices within the vehicle and the outside world, by making both of them accessible to one another. For this purpose, it keeps listening for multicast and broadcast packets on all interfaces, and then, based on the defined rules, these packets are forwarded between the inner and outer interfaces.

- Interface Monitoring - As the name suggests, it keeps overseeing the outer interfaces for events like the plug-in of a new interface, failure detection, and the status of the current interfaces. Interface monitoring is important for keeping the user application informed of the available choices of networks, and for configuring the routing or multicast proxy service.
- GRCBox core - This service is responsible for communicating with applications, validation of the connection rules, maintenance of the database, configuration of the routing service, and managing the multicast proxy service.

**Application interactions with the GRCBox** One of the network interfaces (inner interface) of the GRCBox is used as an access point to which the smartphones and other devices within the vehicle may be connected. This way, the devices can communicate with one another, as well as send and receive data to or from participants of other networks to which the hardware module can reach out to. The choice of the outer interface which is to be used for communication is based on the rules defined by the user applications. A rule is a packet filter that consists of:

- Type of the rule - Rules according to the GRCBox platform are categorised into three different groups namely incoming, outgoing and multicast.
- Name of the interface - Identification of the outer interface to which the rule is to be applied.
- Choice of protocol - Which transport protocol is to be used, for example the Transmission Control Protocol (TCP) [132] or the User Datagram Protocol (UDP) [133].
- Source address - Internet Protocol (IP) address of the source.
- Source port - The source port used for the connection.
- Destination address - IP address of the destination.
- Destination port - The destination port used for the connection.

Thus, based on these parameters, the routing service listens for incoming connections on the outer interface indicated, routes outgoing connections to the specific interface, or forwards multicast packets in both directions between the inner and outer interfaces involved. Note that, for incoming connections, the GRCBox keeps track of the destination address, along with the port to which the incoming data is to be forwarded, and modifies the header of the incoming packets so that the target user device can receive it. However, before an application is able to define the routing rules, it has to register itself with the GRCBox and receive a private secret key for later interactions with the GRCBox. The use of the secret key guarantees that only the rightful owner of a rule is able to renew, modify or remove it from the database. Before a new rule is created, the GRCBox system makes sure that the new rule does not conflict with any of the existing rules; once it is clear that the new rule does not collide with the existing once, it is put into effect and the application is notified. After this, the application is able to initiate the connection that will be routed according to the defined rule. Once the application has finished using the connection, it should notify the GRCBox that the rule is no longer needed, so that it may be eliminated from the system.

**Client support** To allow developers to easily make their applications GRCBox-compatible, we have created a programming language independent remote API, a set of software libraries, and a management application that may be used to define rules so that third party applications may use GRCBox without making any modification in its software code.

- Remote API - May be used to define routing rules remotely.
- Client library - An transparent way to use the features supported by GRCBox making use of the remote API.
- Management application - Allows users of a smartphone to define rules for third party applications; this way, no modification to these applications is necessary. An example of the use of this management application could be configuring a rule to forward traffic related to Voice over Internet Protocol (VoIP) [134] through the interface that provides a connection to the cellular network. Thus, all applications that a user has installed in his or her mobile device, and that are dependent on VoIP data, would be able to connect to the cellular network via the GRCBox, without any sort of changes to their code.

#### 3.1.1.2. Implementation

In this section we are going to present in detail all the different components of the GRCBox that we have touched upon in the previous section.

**Hardware module** Here we are going to describe the physical as well as the software components of the GRCBox device.

- Physical components - As can be seen from the figure 3.2, the GRCBox consists of a controller for processing, a USB hub to connect the network interfaces, and at least two network interfaces: one that is configured as inner, and other as outer interface, along with an optional battery as power source. For the controller, we have chosen to use a cheap embedded computer called RaspberryPi [135], which is compact and of the same size as a credit card, but powerful enough to perform small scale routing. The RaspberryPi uses Raspbian [136], an optimised and adapted version of a general purpose Linux distribution [137], thus supporting most current networking hardware.

- Discovery service - We have used the dnsmasq [138] tool for providing Domain Name System (DNS) [139] and Dynamic Host Configuration Protocol (DHCP) [140] services, thus allowing the user application to discover which node in the network is actually the GRCBox. Thus, when a node connects to the GRCBox, DHCP is used for the configuration of the connection which includes the default DNS server IP pointing to the GRCBox. Because GRCBox also supports DNS, it is unnecessary for the nodes to know the actual IP address of the GRCBox for connection purposes.

- Network Address Translation (NAT) [141] and Routing - The GRCBox uses Iptables [142] to perform Source Network Address Translation (SNAT) on each interface available to the GRCBox, and the Routing Policy Database (RPDB) of the Linux Kernel to redirect the application-defined connections to the chosen interface. For incoming connections, Destination Network Address Translation (DNAT) is performed with the help of forwarding destination address and port number.

- Interface monitoring - To check for information related to the interfaces available to the GRCBox, we have created a set of classes that interact with the nmcli [143] commandline tool, which in turn connects to the NetworkManager [144] system service. This allows us to know the IP address, gateway, type of each interface, and other related information such as whether the interfaces are connected to the Internet.

- Core server - The GRCBox core server is responsible for controlling the different server components, communicating with the client devices, interpreting the client commands, and instructing the component responsible to perform the required action. The core server is based on the REST [145] architecture that defines a way to create, read, update or delete information using HTTP [146], and depends on some constraints and properties. For achieving our goal, we have defined several resources accessible from the

| Resource Name | Syntax | Functions | Description |
|---|---|---|---|
| Root | / | GET | Server status and number of existing rules. |
| List of Ifaces | /ifaces | GET | List of interfaces for outward communication. |
| Iface | /ifaces/{*iface-id*} | GET | Information related to a specific interface. |
| | | POST | Configure interface parameters. |
| List of Apps | /apps | GET | List of all registered applications. |
| | | POST | Register a new application and generate a secret key. |
| Application | /apps/{*app-id*} | GET | Information related to an application. |
| | | POST | Keep the application registered in the system. |
| | | DELETE | Remove an application and related rules from the system. |
| List of Rules | /apps/{*app-id*}/rules | GET | Returns all rules an app has registered with the system. |
| | | POST | Register a new rule for managing data traffic. |
| Rule | /apps/{*app-id*}/rules/{*rule-id*} | GET | Information about a rule. |
| | | DELETE | Remove a single rule from the database. |

*Table 3.1: GRCBox REST API.*

inner network that allows clients to register a new application, check the status of the GRCBox, and retrieve information about the active interfaces with the help of the Java framework called Restlet [147]. More details about the REST API can be found in table 3.1.

- Multicast and broadcast proxy - The GRCBox supports sending and receiving of broadcast and multicast messages between the inner and outer networks. This proxy has been implemented using RockSaw [148], which is a Java [149] library that allows the use of raw sockets, including defining the low-level contents of IP packets. For every new multicast or broadcast forwarding rule, a new proxy instance is created.

**Client library** To facilitate the development of applications that can take advantage of the GRCBox, we focused our attention on promoting a library for mobile devices. For the first version we have concentrated on Android [150] devices owing to their huge popularity and market share. The different components of the Java library for Android-based devices is presented in table 3.2.

| Class Name | Description |
|---|---|
| GrcBoxSocket | An extension of the Java Socket Class. |
| GrcBoxServerSocket | Extends the ServerSocket Class in Java. |
| GrcBoxDatagramSocket | Java DatagramSocket Class is extended. |
| GrcBoxMulticastSocket | Adaptation of the Java MulticastSocket Class. |
| GrcBoxClient | Class that manages the registration. |

*Table 3.2: GRCBox Client Library.*

### 3.1.1.3. Use case and Performance Analysis

In this section, we are going to take a detailed look on how an application connects via the GRCBox with the help of an example, and also present some preliminary results obtained in our experiments with the GRCBox.

**GRCBox use case** Let us assume that we have an application installed on a mobile device that is connected to the GRCBox, and it wants to access a website, say google.com, through the station mode wifi interface, even though a different interface is configured as default. To understand the steps to follow for this action, we invite taking a closer look at figure 3.3.

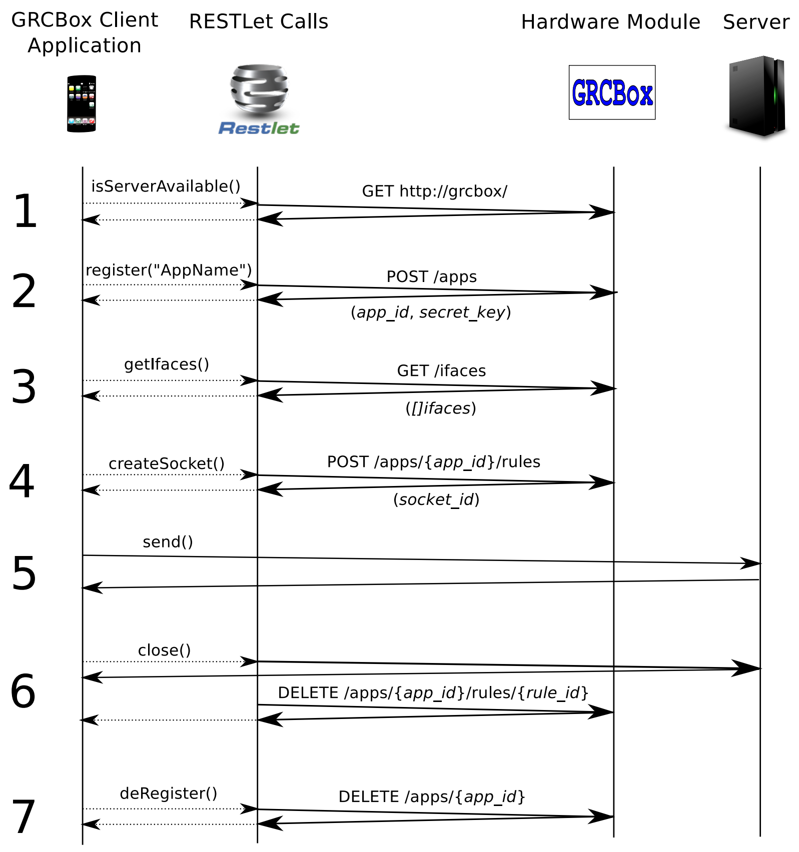


*Figure 3.3: How an application connects making use of the GRCBox.*

- Step 1 - The client device has to be connected to the inner interface of the GRCBox, which is configured to work in access point mode; then the application has to create an instance of the already available GRCBoxClient class. It later checks if the GRCBox is functional, which is done by calling the GRCBoxClient.isServerAvailable(). If no response if received, the application can choose to disconnect from the GRCBox, and use the standard networking libraries.

- Step 2 - Upon making sure that the GRCBox is functioning properly, the application must register itself with the GRCBox by calling GrcBoxClient .register(*"AppName"*) to use the features supported by it. After successful registration, a new thread is started to send periodic keep-alive notifications to the GRCBox. This helps us detect disconnections, in which case the privileges and networking rules that the client application was enjoying can be revoked.

- Step 3 - Next, the application needs to decide which outer interface available to the GRCBox it wants to take advantage of, for future communications. An application can switch between interfaces at any time. Hence, it retrieves information related to the available interfaces by making a call to the method GrcBoxClient.getInterfaces(), and chooses the one that is most apt for its purpose.
- Step 4 - The next step involves the creation of sockets for sending and receiving data. The function GrcBoxClient.createSocket(*dstAddr*, *dstPort*, *iface*) is used for this purpose to create the socket and the routing rule associated with this socket.
- Step 5 - Once the socket has been created, the application can send and receive data through the chosen interface of the GRCBox.
- Step 6 - The application, upon finishing the data exchange with the remote host, should close the GrcBoxSocket, which will also remove the routing rule from the system.
- Step 7 - Finally, before the application stops working, it should deregister itself from the GRCBox. This causes the keep alive thread to stop communicating with the GRCBox, and all information related to the application is eliminated from the GRCBox database.

**Performance Analysis** We have performed some preliminary experiments to analyse the different aspects related to GRCBox, like the delay introduced in the setup of routing rules before the actual communication can begin, the delay that the data suffers when travelling through the GRCBox, and the bandwidth that the current version of the GRCBox can offer to different client applications.

- Control delay - In this category, we consider the delay involved while checking the status of the GRCBox, retrieving information related to the interfaces, performing registration and de-registration of the application from the GRCBox system, and also the time taken to set up and delete a routing rule from the system.

  Table 3.3 shows the time taken to perform the actions supported by the GRCBox when a smartphone application connects and executes the supported functions. Note that, even thought delay is involved when performing these operations on the GRCBox, these actions, in most cases, are performed only once during the entire application life cycle, and hence do not significantly affect the performance of the application. Also, upon close examination

| Action | Average Delay (s) | Conf. interval (95 percent) |
|---|---|---|
| GRCBox status | 1.55 | 0.22 |
| Interface information | 1.25 | 0.03 |
| Register application | 0.96 | 0.03 |
| De-register application | 1.22 | 0.02 |
| Register rule | 1.29 | 0.03 |
| De-register rule | 0.21 | 0.02 |

*Table 3.3: GRCBox control delay.*

of the results presented, we can see that the confidence interval for checking the status of the GRCBox is too large, because the monitoring module sometimes block when the system is updating information related to the interfaces, which may take as long as 10 seconds. A possible solution is the use of the D-Bus [151] interface, which employs shared memory objects, and also provides a subscriber-publisher interface rendering polling unnecessary.

- Hop delay - Using the GRCBox for connection purposes increases the length of the path that data travels by one hop. So, we now proceed to take a look at two cases: the first case being the download time for a 5 Mb file, and for the second case we ping a website. Both the cases were tested with and without the GRCBox for comparison purposes.

| Connection | Average time | Conf. Int./ Std. Dev. |
|---|---|---|
| | 5 Mb file download | |
| Using GRCBox | 6.34 s | 0.3 (95% C.I.) |
| No GRCBox | 7.00 s | 0.4 (95% C.I.) |
| | ping www.upv.es | |
| Using GRCBox | 16.96 ms | 23.97 (Std. Dev.) |
| No GRCBox | 14.21 ms | 35.90 (Std. Dev.) |

*Table 3.4: GRCBox hop delay.*

Table 3.4 shows the results. We can see that, in both the cases, there is not much difference in the time taken to download a file or to ping the website `www.upv.es` when using the GRCBox, or when connected directly to the nearest network. However, we can see that, for our ping experiments, the standard deviation was very high, pointing towards some network instability.

- Bandwidth - We used the iperf tool to evaluate the maximum throughput supported by the GRCBox for UDP and TCP connections. The results are presented in figure 3.4, and involve two different scenarios: In scenario 1, the two smartphones used for data exchange were connected to the same GRCBox device; while in the scenario 2, the same devices were connected two different GRCBox units, which in turn communicated with each other making use of the adhoc network.

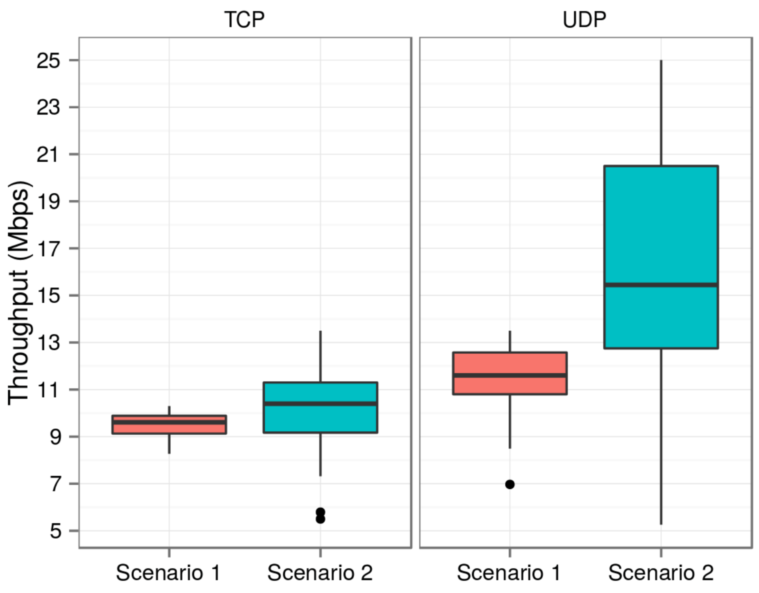


*Figure 3.4: Throughput offered by the GRCBox.*

From the figure 3.4, we can see that the GRCBox is capable of providing a mean bandwidth of 10.5 Mbps for TCP traffic, and 15.5 Mbps for UDP traffic, although the worst value for both TCP and UDP was close to 5.5 Mbps.

### 3.1.1.4. Conclusion: GRCBox

We have presented a low cost connectivity device named GRCBox that supports the complete integration of smartphones to VNs. The GRCBox unit consists of a hardware module that is to be placed within the vehicle for creating the network of vehicles, and it also comes with a set of libraries to help developers build applications that can take advantage of the communication that the GRCBox has to offer. Although the presented solution has been tested and validated, it suffers from slow monitoring of the interfaces installed, so we want to switch to using D-Bus for better performance. Also, we want to develop a smartphone application that would allow more flexibility and better control of the GRCBox, like being able to reboot it whenever required, and establishing the default outgoing interface.

## 3.2. Android Applications

The principal idea behind the development of the Android Operating System (OS) was to let developers freely modify, invent and implement drivers or other features. It consists of five different layers, namely the Application Framework, the Binder Inter-Process Communication (IPC) Proxies, the Android System Services, the Hardware Abstraction Layer (HAL), and lastly, the Linux Kernel.

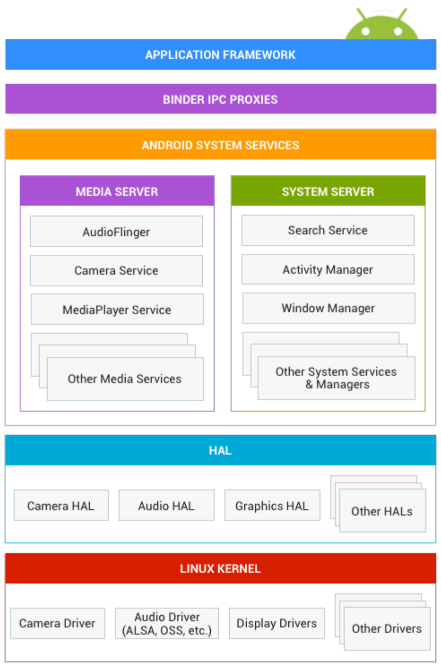


*Figure 3.5: Structure of the Android OS [6].*

Figure 3.5 shows the different layers. The Application Framework is used by the developers of Android. It is based on the multi-user concept, where every single application is considered as a different user. Android applications are developed in Kotlin, Java, and C++ languages, and the developed application is compiled by the Android Software Development Kit (SDK) tools to create an archive file known as Application Package Kit (APK). The APK file thus contains the application to be installed, along with any data or resource it might require to operate. During installation, each application is assigned a unique user ID that is only known to the system, which assigns the necessary permissions for the files in the application so that they are only accessible by it. Furthermore, by default, each application runs on its own Linux process, and each process has its own Virtual Machine (VM), so that it runs isolated from other applications. A process is started when any of the application components need execution, and it stops when it is no longer needed or to make memory available. Applications can share a Linux user ID when they need to run on the same process, or access each other's files.

The Binder IPC helps applications to communicate with one another, and it is sourced through the Kernel of the Android OS. This helps to avoid unnecessary allocation of space unlike other IPC mechanisms. Also, it aids in running multiple processes at the same time on a concurrent level, and enables the high level framework to communicate with the Android services.

An Android System Service is compiled code launched by a user of the system. It runs in the background without providing any interface, and keeps on executing even if the application is closed. It may be of two types: Started and Bound Services. Started services are launched when applications call it, and usually performs a single operation without returning anything. Regarding bounded services, they offer a client-server based interface and runs as long as the foreground application is running. The Android system services provide the necessary information to the user application, thus helping them to work properly, and here the Binder IPC comes in handy to maintain communication between the services and the user application.

The HAL is specially designed for vendors to insert functionality without having to modify the system. It consists of two structures: the Module and the Device. The HAL module structure is stored as a shared library with extension *".so"* format, containing metadata information such as version number and author. While, the device structure is the actual hardware, and defines a more comprehensive version of the generic hardware information, like pointers and other information specific to the hardware. In Linux-based systems, an application communicates with the hardware using system calls, but in Android, this is achieved via Java APIs.

Lastly, the lowest layer is the Linux kernel, which is similar to a basic Linux OS, but it comes with advanced features like wakelocks, double-tap to unlock, or support for other similar features related to mobile devices. Unlike the basic Linux, these features are important since the kernel is aimed for portable devices, and thus it needs to include aggressive memory and battery management techniques.

We already know that each application in Android uses a separate VM. It is important to note that, until 2014, Android release of version 5.0 Lollipop [152], Dalvik VM [153] was the default system. Which is a register based VM specially designed for constrained environments with low memory and processor speeds, like embedded systems. It simply uses designated memory locations to simulate the functionality of registers in microprocessors. Thus, Android applications are first compiled to *".class"* files, which are then translated to Dalvik bytecode and combined together into a single executable (DEX) file. This bytecode is loaded and interpreted by the Dalvik VM. To optimise this process, during the first application launch an optimised DEX file is created, but this file is not portable between devices or Dalvik VMs. With the introduction of the Android Runtime (ART) [154], which is the predecessor of the Dalvik VM, and is backward compatible to some extent, ART employs Ahead-of-time (AOT) compilation and improved garbage collection techniques to enhance performance. Some benchmarks [155] show performance improvements of about 10% in some cases, despite slowdowns in case of other applications.

### 3.2.1. Contribution II: Messiah

Navigation applications are very popular among drivers due to their usefulness when it comes to reaching a destination using routes that one is little familiar with. Our goal is to develop an Android application that, apart from providing navigation, would warn drivers of important emergency vehicles approaching on the map, along with their possible future route. This way, drivers can react according to each specific situation. For example, if a driver is alerted of an approaching ambulance long before the siren is heard, it gives him or her more time to react and leave the lane vacant for the ambulance. If instead, the route of the ambulance is made known to the driver, he or she might choose to avoid that particular route, and thus contribute towards improving the response time of Emergency Medical Services (EMS).

Our application is based on OsmAnd [156] which uses the free OpenStreetMap (OSM) [157, 158] data. OSM is dependent on a community of mappers, thus emphasising on local knowledge. The contributors use aerial imagery, GPS devices and low-tech field maps to commit towards the maintenance of map data. The difference between our application and a traditional navigation-based application is that our application communicates with other users of the same application, and exchanges information to display on-screen the position and future route of important approaching vehicles.

#### 3.2.1.1. Features of the Messiah Application

Our application, named Messiah, aims to display on a map view the presence of priority vehicles like ambulances, police cars or fire brigades in the vicinity of the common vehicle. Vehicular communication is used towards making users aware of the tentative future route of these emergency vehicles, apart from displaying their current position on the map. Thus, based on this information, drivers can prepare before-hand, and this gives them more time to take decisions, rather than having to wait until the EMS vehicle appears on the rear view mirror, or its siren is heard. Drivers may decide whether to make space for the emergency vehicle and let it pass, or try to completely avoid the already known route of this important vehicle. The main reason to develop this application was to improve the response time of these emergency vehicles, and in turn help reduce casualties that might result from traffic congestion. Hence, cities with dense traffic would largely benefit from this application due to two clear reasons: firstly, the large number of vehicles participating in the network would facilitate the exchange of application data; and secondly, the emergency vehicles would be able to reach their destination taking advantage of the streets that would normally be crowded. Of course, the success of the designed application largely depends on how responsive and responsible the users of the application are.

Another very important functionality that our Messiah application has to offer is that it provides users the ability to request for help when in critical situations, and while the lives of the passengers within the vehicle are at stake. This feature could come in handy when an accident occurs; in that case, with the mere click of a button, users can disseminate a request for aid to nearby vehicles, taking advantage of the vehicular network. Other vehicles in the vicinity, upon receiving this request comes to know of the exact location of the vehicle in need of help. This way, it may rush to the aid of the vehicle requesting for help, or call the local emergency number, even if the vehicle requiring assistance is not directly within the line of sight. Thus, our drive safety application designed for Android devices takes advantage of communication among vehicles in order to help make a more efficient and safer use of the road network, potentially saving lives.

Users of the application are classified into three different groups or categories, which strongly reflects the mode in which the application is functioning. The different modes of the application are as follows:

- Civil Mode - Participants of the network which are common vehicles, make use of this mode in the application.
- Administrative Mode - To be used only by on-duty emergency vehicles, like police cars, ambulances and fire brigades.
- SOS Mode - Vehicles in danger of any kind can request for help making use of this mode of operation.

When the application is first launched, it starts providing just the navigation related services to the users, hence minimising the information displayed on screen

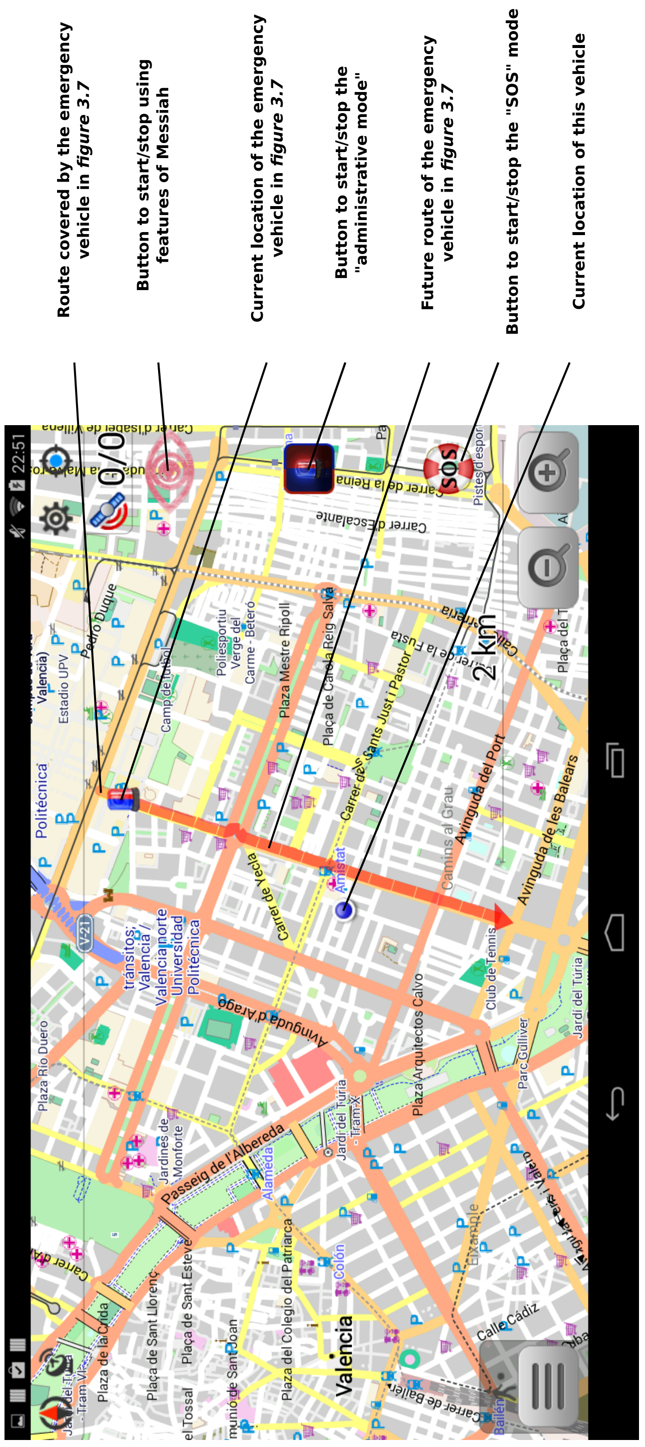

Route covered by the emergency vehicle in *figure 3.7*
Button to start/stop using features of Messiah
Current location of the emergency vehicle in *figure 3.7*
Button to start/stop the "administrative mode"
Future route of the emergency vehicle in *figure 3.7*
Button to start/stop the "SOS" mode
Current location of this vehicle
0/0
SOS
2 km
22:51
Valencia

*Figure 3.6: Device working in "Civil Mode".*

to reduce driver distractions. The users, when using the application, may choose to only take advantage of its navigational capabilities, without being obligated to participate in the “drive safety” network that the application is capable of creating. Others who wish to cooperate and actively engage in the “drive safety” network can do so merely by clicking a button available on screen. On doing so, the application starts functioning to its full potential, and it assumes that the current vehicle is a common civilian vehicle, consequently beginning to work in the “Civil mode”. In this mode, it receives or forwards data to and from its neighbouring nodes using V2V communication. Apart from this, it also displays on screen any received information, that is, the location of any nearby emergency vehicles and nodes in need of assistance. In case of the presence of an emergency vehicle with a predefined route in the vicinity, all or part of its route, along with its current location, is displayed on screen in the vehicle participating in the “Civil mode”.

Figure 3.6 shows the different options available on screen when a participating node is working in the “Civil mode”. We can see that, on the right hand side of the screen, there are three buttons: one to start or stop participating in the “drive safety” network, the second to switch to the “Administrative Mode”, and lastly, a button to switch on or off the “SOS Mode”, that lets users call for help. The application shown here, which is currently working in the “Civil mode”, displays the current location of the vehicle itself, which is using our application. Although the current vehicle has no route configured in this case, it is nevertheless possible to do so, thus fulfilling the purpose of a navigation application. Next, we see on screen the location of an administrative vehicle present close by, along with the route that it intends to follow in the near future. Note that, even though in this example we observe only one vehicle in the area which is an on-duty emergency vehicle, in real scenarios, if more than one administrative vehicle is present, all of them would be displayed on screen to all vehicles within a certain range, irrespective of the mode in which they are functioning. Thus, in short, the “Civil mode” is the most basic of the three modes in which the application can function, which is sending and receiving application-related data, while displaying relevant information on screen. In all other modes, the application apart from doing the same as in the “Civil mode”, has some added responsibilities.

An on-duty emergency vehicle may choose the “Administrative mode” of operation. Although, no security checks have been implemented so far to validate the identity of the users of this mode, in the future we intend to make this option available only to authorised personnel operating a police car, ambulance or fire brigade. In this mode, a device with the Messiah application installed, performs the forwarding of data received from other nodes, and displays relevant information on screen, as in case of the “Civil mode”. Thus, a user in the “Administrative mode” is also aware of the location of other emergency or “SOS” vehicles in the area. Apart from this functionality, it also broadcasts its current location, route and destination information to other vehicles in its vicinity. We limit the forwarding of messages up to 1 km radius from the location where the message was first generated. This serves two purposes: it limits broadcast storms and restricts circulation of unwanted stale information, because it would take more time for messages to reach destinations further away, in which case the source of the information could have already changed its position, thus rendering the information propagated useless. We discard the use of time-based deadlines since time is generated by each individual system, and clocks are not synchronised in our case.

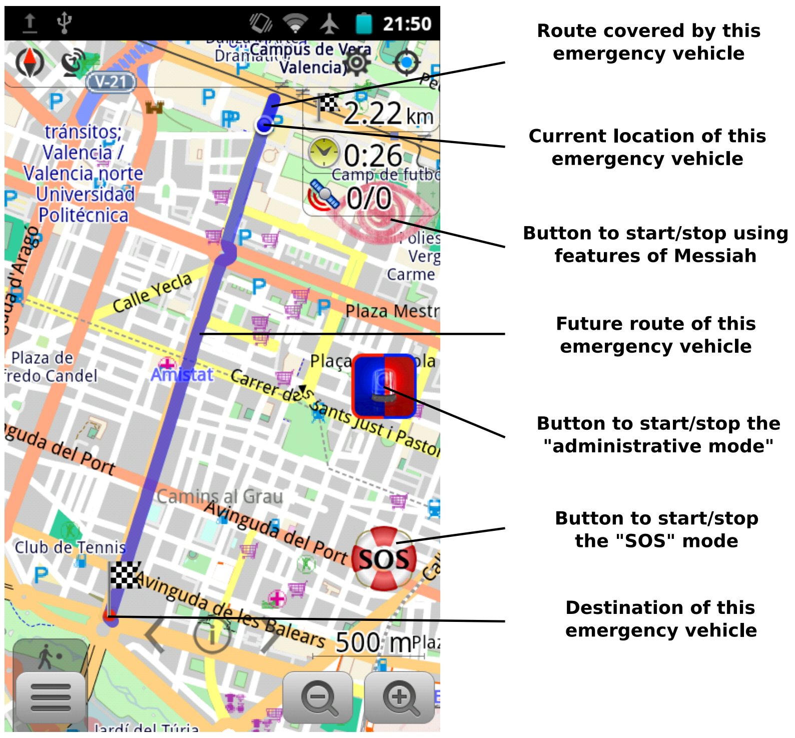

21:50
2.22 km
0:26
0/0
SOS
500 m
Route covered by this emergency vehicle
Current location of this emergency vehicle
Button to start/stop using features of Messiah
Future route of this emergency vehicle
Button to start/stop the "administrative mode"
Button to start/stop the "SOS" mode
Destination of this emergency vehicle

*Figure 3.7: Device working in "Administrative Mode".*

Figure 3.7 shows a device working in the "Administrative Mode". In this example, the current node is an active emergency vehicle with a predefined route and destination, that is shared with the neighbouring vehicles by taking advantage of the vehicular network. The devices receiving this information first check if they had already received a copy of the message previously. If the information received is new, they calculate if the distance between the source of the message and themselves is less than equal to 1 km, in which case the message is forwarded to the neighbouring nodes, and relevant information is displayed on screen. In the example we present here, the node in figure 3.7 broadcasts its location and route information, which is received by the device in figure 3.6, and the obtained information is displayed on screen. Note that, if the future route of the emergency vehicle is too long, the whole route is not sent, but only a part of it. In figure 3.6 we can see that this particular node receives only a part of the route of the vehicle shown in figure 3.7, this route is displayed with the help of a broken line and an arrowhead to indicate the future route. The arrowhead indicates the possible direction in which the route will be updated next.

Another option supported by the Messiah application is the "SOS Mode". This mode enables users in distress to request for rescue. It is similar to the "Administrative Mode", the only difference is that vehicles using the "SOS Mode" broadcast their current location without including the future route. It is assumed that vehicles in distress have faced some kind of accident, or have been rendered immobile. Upon receipt of a request for aid from an vehicle in distress, the nearby vehicles display a special icon on screen denoting the current location of the vehicle requiring assistance. Note that disconnections are common in high mobility scenarios, and that the Messiah application is capable of detecting disconnections of such type. Also, the application continues to function even when it is not on the foreground, in that situation no information is displayed on screen, but the application keeps functioning in the background by sending and forwarding data as necessary.

#### 3.2.1.2. Implementation Details of Messiah

The main components of the Messiah application are the Messiah service, a Communicator thread, an IdGenerator class, the MessageGenerator class, the Locator class, and a DatabaseManager class. The Messiah service is the backbone of the application, and it controls the different other components. It is launched when the user chooses to join the "drive safety" network, and generates a unique identification number for the node with the help of the IdGenerator class. It collects information related to the mode of operation of the application, e.g. if the current node is an ordinary or emergency vehicle. If the application is operating in the "Administrative Mode", the Messiah service also collects route-related information from the Map Layer. Later, it starts the Communicator thread, which takes care of receiving, sending or forwarding application data. The other classes, like the MessageGenerator is responsible for the generation of application-compatible messages in case some information needs to be sent. This class is also used to detect if received messages are actually relevant to the application. The Locator class, on the other hand, maintains up-to-date information regarding the positional knowledge of the device. Instead, the DataBaseManager class acts as a cache for incoming messages, and it is used to detect duplicate messages. Figure 3.8 shows the different working components of the Messiah application, and how they interact with one another.

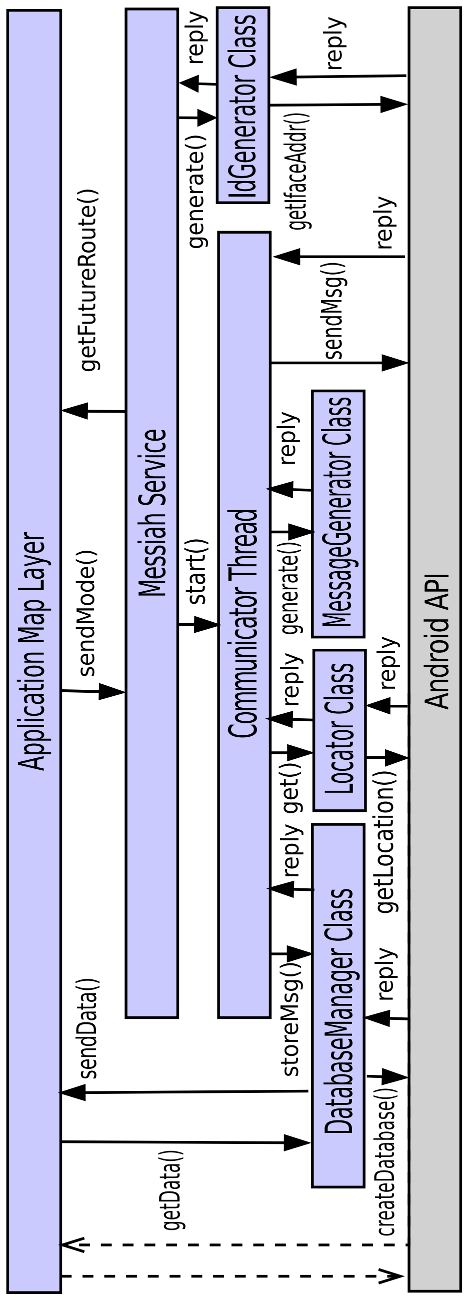


*Figure 3.8: Architecture of the Messiah Application.*

| ID | TimeStamp | Type | Location | Future Route | Destination |
|---|---|---|---|---|---|

*Figure 3.9: Contents of messages used by the Messiah Application.*

Figure 3.9 shows the structure of a message used in the application. It is composed of the following fields: node-id, time stamp, type of message, current location of the sender, its intended future route, and its destination. Let us take a look at each of these fields in more detail:

- ID - Contains the unique node id generated by the IdGenerator class.
- TimeStamp - A time stamp to indicate the time of message generation by the sender.
- Type - Messages used by the Messiah application can be of three types: Administrative, SOS or Bye-Bye. This field indicates which type among the three it belongs to.
- Location - The location of the sender at the time of message generation.
- Future Route - A list of GPS locations that the sender is supposed to follow.
- Destination - Geographical position that denotes the end of the sender's trajectory.

As stated earlier, packets used in the application have the same format, but they can be of three distinct types: i) Administrative, ii) SOS, and iii) Bye-Bye. The data contents of messages for vehicles operating in "Administrative Mode" and "SOS Mode" lets neighbours know about the current location of the sender and its future route, if any. Instead, a "bye-bye" type of message is used when an active vehicle working in the "Administrative Mode" or the "SOS Mode" has completed its function, and wants to remain attached to the network but as an ordinary vehicle; in other words, this type of message is used when a vehicle previously working in a different mode wants to switch to the "Civil mode".

Figure 3.10 presents the steps involved in the packet generation and sending process, which occurs when the application is running in the "Administrative Mode" inside an emergency vehicle, or while in the "SOS Mode" for a vehicle requiring assistance. The Messiah service requests the current location of the device from the Locator class, checks the mode of operation, and, if a route has been configured by taking advantage of the navigational capabilities of the application. All this information acquired is passed onto the Communicator thread, which prepares the packet to be sent with the help of the MessageGenerator class, allowing

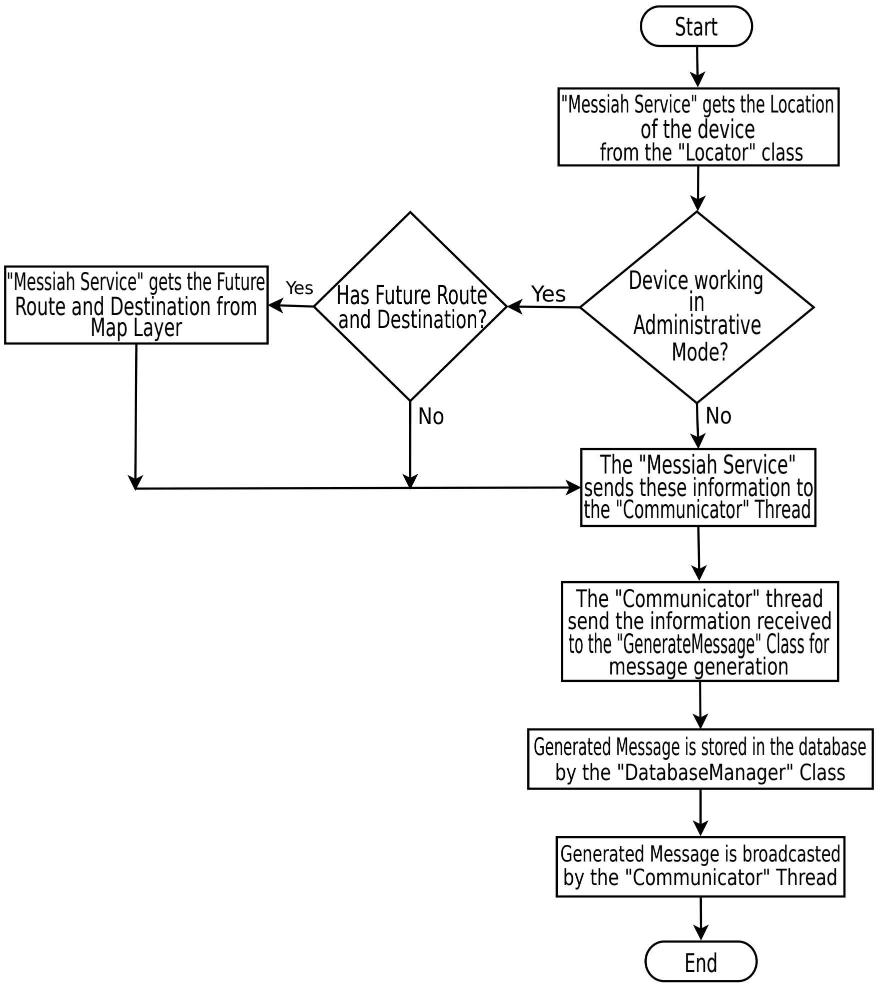

Start
"Messiah Service" gets the Location of the device from the "Locator" class
Device working in Administrative Mode?
Yes
Has Future Route and Destination?
Yes
"Messiah Service" gets the Future Route and Destination from Map Layer
No
No
The "Messiah Service" sends these information to the "Communicator" Thread
The "Communicator" thread send the information received to the "GenerateMessage" Class for message generation
Generated Message is stored in the database by the "DatabaseManager" Class
Generated Message is broadcasted by the "Communicator" Thread
End

*Figure 3.10: Message generation and sending in Messiah.*

this packet to be broadcasted. Before sending the message, a copy of it is retained locally, using the DatabaseManager class.

Similarly, figure 3.11 depicts how the application behaves upon receiving messages. Note that all the three different modes ("Civil Mode", "Administrative Mode", and "SOS Mode") in which the application functions operate in a similar fashion on receiving an incoming message. The Communicator thread constantly keeps checking for the arrival of messages. When some data is received by the Communicator thread, it scrutinises whether the received data is a valid message by consulting with the MessageGenerator class. On confirming the validity of the data received, the location field of the message which contains the geographical position of the source during message generation, is accessed. This is compared to the position of the receiver node to verify if the source is within a 1 km radius; otherwise the message is discarded, and any entry in the database related to that same source is also eliminated with the help of the DatabaseManager class, before updating the display for the users. Next, the type of message received is enquired. Remember that a message used by the Messiah application can be of three types: Administrative, SOS, or Bye-Bye. If the message is of the Bye-Bye type, it means that an emergency or SOS node has completed its operation, and is switching to the "Civil Mode". In that case, the database is audited for any entries from the same source which generated the Bye-Bye message. If a database entry has been encountered, and only in that case, the message is considered to be of importance and is re-broadcasted before updating the on-screen display; otherwise, it is discarded. For the other types of messages, namely Administrative and SOS, the information is updated in the database if previous entries from the same source exist; otherwise a new entry is added. A local time stamp is appended to the information received in the message, which aids in the detection of connectivity losses, and the elimination of stale entries. Finally, the received packet is re-broadcasted without any changes made to it, and the user display is updated accordingly.

#### 3.2.1.3. Prototype Deployment and Evaluation

The Messiah application relies on communication among vehicles to function. Hence, for deployment purposes, we used two cars equipped with the GRCBox (discussed in section 3.1.1) to provide V2V communication. Within each car, an Android device with our application installed was placed so that it could benefit from the GRCBox. Using this setup, we performed experiments to analyse the behaviour of the application in scenarios with various levels of obstacles that could make communication difficult.

In figure 3.12, we have considered three different scenarios, with two cars located at an intersection on different roads. The first scenario considered was a populated residential area with narrow roads and tall buildings; the two vehicles were at line of sign only at the intersection, thus experiencing maximum packet

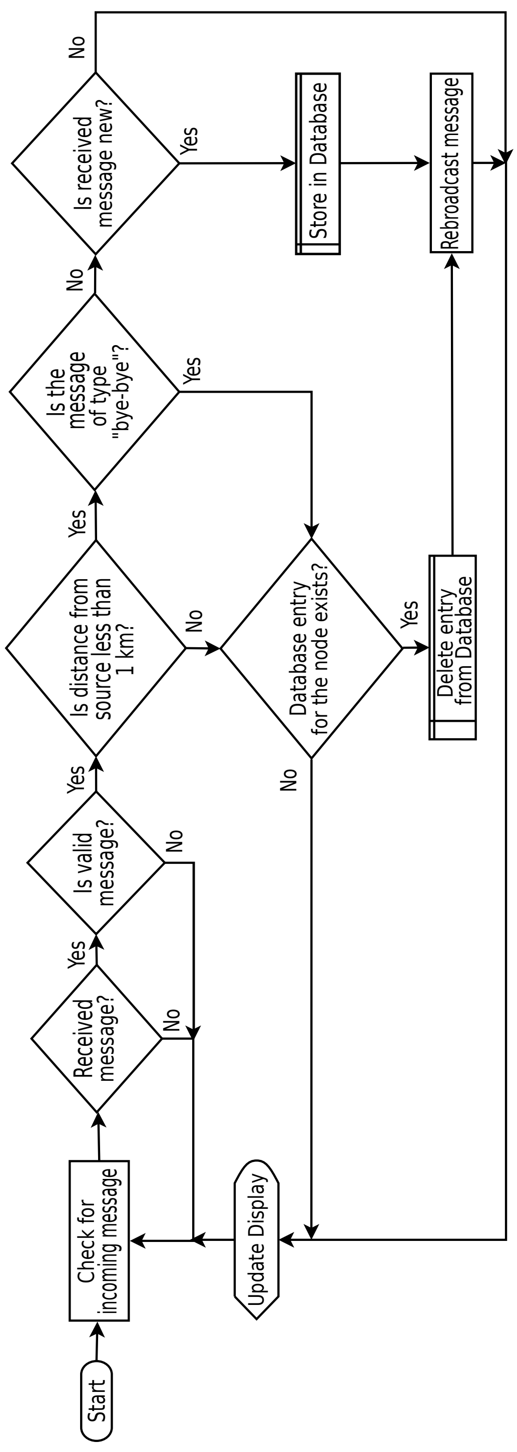

Start
Check for incoming message
Received message?
Yes
No
Is valid message?
Yes
No
Is distance from source less than 1 km?
Yes
No
Is the message of type "bye-bye"?
Yes
No
Is received message new?
Yes
No
Store in Database
Rebroadcast message
Database entry for the node exists?
Yes
No
Delete entry from Database
Update Display

*Figure 3.11: Receiving and forwarding of messages.*

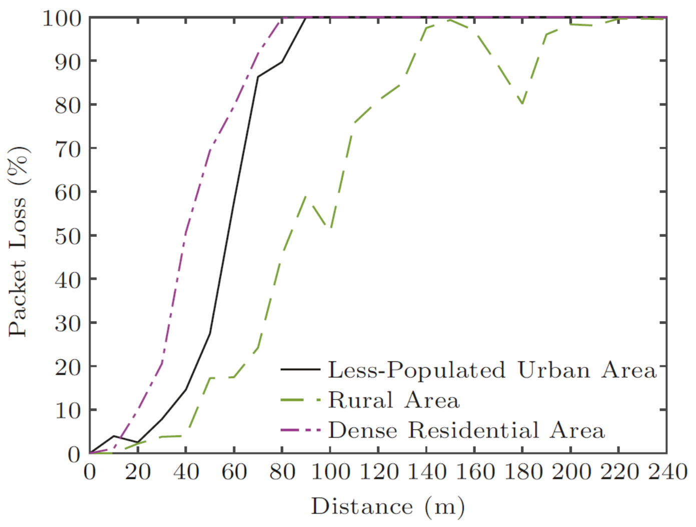


*Figure 3.12: Packet loss with distance using the Messiah application.*

loss levels. The second scenario was a less crowded urban area with wider roads, thus suffering from less losses compared to the first scenario. And, as expected, we observe that the best performance is achieved in the rural area, where nearly no obstacles were present. Thus, to take advantage of the Messiah application in urban scenarios, we would require a network of densely populated nodes.

#### 3.2.1.4. Conclusion: Messiah

We have introduced an application that provides navigation and emergency notifications, aiming to reduce congestion along the route of emergency vehicles, and ultimately resulting in reduced response time. It helps to create a safer traffic environment as it displays the position of important vehicles like police cars and ambulances on the navigation screen, thus warning drivers to drive with attention and aid in taking decisions based on the information displayed. It also includes a special mode of operation that a vehicle in distress may use to call for help. In this mode, a exclusive icon is visible on screen of the vehicles that are close to the "SOS vehicle", and hence they can rush to its aid.

### 3.2.2. Contribution III: Forward Collision Warning

Warning generation upon the detection of a high probability of collision has always been a topic of interest among researchers. Early works include [159], which studies the use of radar technology for detecting possible collision. An effort by General Motors [160] resulted in the publication of [161], dedicated to the design of a simulation tool to evaluate technical and functional specifications of FCW systems based on radar sensors. A beacon-based collision warning system [162] was designed by Miller et al. in 2002 that did not require vehicles to be in line of sight for operation.

One of the first vision-based approaches was a simple algorithm to detect vehicles, and warn drivers when they were too close to the other vehicle, was presented by Srinivasa [163]. An improved version [164] combining the use of data from a forward looking camera and a radar was proposed by the same author a year later. Other approaches that rely on data from a camera for warning generation was presented in [165], [166] and [167]. [168, 169] is an investigation initiated by Volvo [170] on collision avoidance and automatic braking by only making use of a car mounted with a radar and camera. Another application that uses a vision based approach, depending on Sobel edge enhancement [171] and an optical flow algorithm, is [172]. Chang et al. [173] studied the fusion of vision and GPS sensing for collision warning.

Misener et al. [174], on the other hand, described a cooperative collision warning project that included forward collision warning, an intersection assistant, along with a blind-spot and lane change assistant. First, each vehicle estimates its position by combining the data from GPS with information like wheel speed, steering angle, and yaw rate. Later, vehicles exchanged this information among themselves to generate warnings as required. A similar application based on the use of GPS and motion sensors was presented in [175]. Fusion of data from Light Detection and Ranging (LiDAR) and other on-board sensors that compute vehicle speed, acceleration, and brake signals is used in [176], presented by Lei et al. Other related studies, like [177], show the positive effects of vibrotactile feedback on the steering wheel and seat-belts for FCW systems.

Thus, we have come across sensor-based solutions that aim to generate warnings when the probability of collision is high. These solutions are mostly vision based, although there are others that rely on GPS, radar technology, or LiDAR. One thing these solutions have in common is that they are dedicated systems, and only applicable to the scenarios they were designed for. Our aim is to build a FCW application for a smartphone-based testbed which can be reused for taking advantage of other ITS applications developed for the same platform. The designed FCW application uses vision and location data to not only alert the user of the application, but also the driver of the car in front of a probable accident. The application relies on V2V communications for this purpose.

#### 3.2.2.1. Application Features

The FCW application that we will present in this section is based on OpenALPR [178], and has been designed to be used on bidirectional two-lane roads in urban areas. It is aimed at Android based smartphones or devices that posses at least a back camera and GPS. The device is to be placed on the dashboard with the camera facing ahead, as the application aims to detect vehicles in front that are too close to the current vehicle using license plate recognition. On detection of the plate, making use of the back camera of the Android device, we can calculate the distance of the vehicle from the one in front since license plates have a fixed size that depends on the geographical region. Note that, if more than one license plate is detected in the frame, meaning there are multiple vehicles present, distance is only calculated for the vehicle directly ahead travelling in the same direction. Thus, we need to select only the plate of the vehicle ahead, and discard the others.

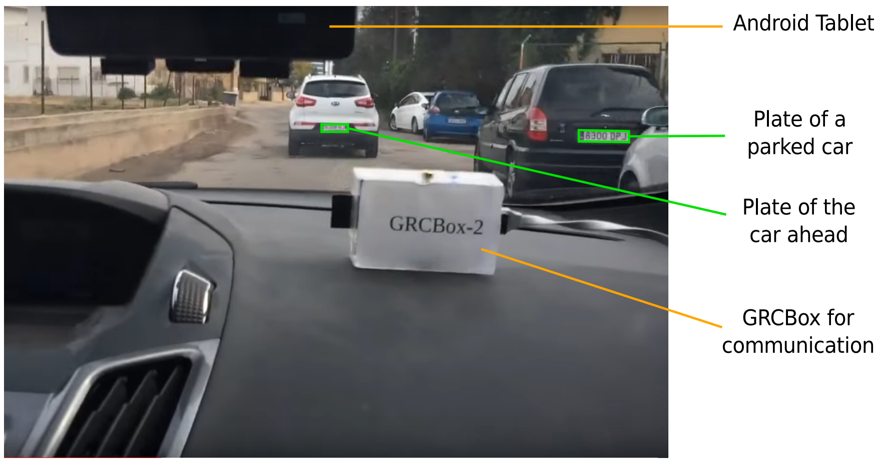


*Figure 3.13: Experimental setup when testing our application.*

Figure 3.13 shows one of the photographs taken during our experiments with our Android FCW application. It demonstrates the complete setup consisting of a dashboard mounted Android device with GPS and back camera, and a GRCBox that allows inter-vehicular communication. The GRCBox is equipped with at least two wireless network interfaces: one configured to act as an access point, and the other one in ad-hoc mode; the device within the car with the application installed is connected to the wifi access point provided by the GRCBox. The GRCBox forwards all data received from one interface to the other acting as a router; this way the application is able to take advantage of the vehicular network that is created using the interface working in the ad-hoc mode. We can see from the figure that the license plate of a parked vehicle and the plate of the vehicle travelling ahead has been recognised. Cases might arise where plates of vehicles coming from the opposite direction could also be recognised. Thus, to detect and eliminate such cases, we take advantage of GPS data exchanged among cars using V2V communication.

Once cases like recognised plates of parked vehicles, and those of vehicles coming from the opposite direction have been eliminated, only the license plate of the vehicle travelling ahead remains. The distance between the vehicle behind and the one ahead is calculated based on the size of the plate in the image captured. If this distance is less than the average length of a car, then alerts are generated. Also, the vehicle ahead is informed of the situation using the GRCBox network.

#### 3.2.2.2. Functional Details of the Application

The different steps that the application follows to detect situations when cars are too close to one another and generate alerts are: broadcast of GPS data and other application-related data by the vehicles; recognition of the license plate of the car from the image gathered using the camera of the device, in which the application is installed; selecting the plate of the vehicle ahead among all recognised plates; estimation of the distance between the vehicle ahead and the one behind; finally, alerting the drivers of both vehicles if safe distance is not maintained. Now, let us look each of these steps in more detail.

- Vehicles broadcast message - As soon as the application is first started, the user needs to introduce the vehicle plate number in which the device would be placed and used. Later, the application checks for a location fix and GRCBox connectivity, and starts broadcasting messages as soon it has acquired two different location fixes a couple of seconds apart. The two location fixes, consisting of the current and previous locations of a device, are constantly updated while the application keeps running.

  |  |  |  |  |  |
  |---|---|---|---|---|
  | Id | Time Stamp | Sender IP | Current Location | Previous Location |

  *Figure 3.14: Structure of the packets used in our FCW application.*

  A vehicle while broadcasting data, simultaneously listens for incoming messages from other vehicles within its communication range. The messages used by our application, as can be seen in figure 3.14, contain a message id, which is the license plate number of the vehicle, the sender generated time stamp, the IP address of the sender, the current location, and the previous location of the sender. The broadcasting of messages is only stopped when the application is closed by the user on terminating the journey. Note that only the vehicles within a one hop radius will receive the message, as no re-broadcasting or forwarding of messages takes place. The received message is saved by the application in its database, adding a local timestamp. Entries in the database are overwritten when new packets are received from the same source, and eliminated if no update is received from the same source for a long time.

- Plate recognition - While the communication between devices using our application is happening in the background, at the same time the back camera of the smartphone or tablet is used to capture images, and processed to recognise license plates within the image, if any. For this purpose, the following sequence takes place: detection of plate regions, converting regions of interest into monochromatic images, character analysis, edge detection, de-skewing, segmentation of characters, Optical Character Recognition (OCR), and generation of the result based on region-wise known templates. Let us take a look at each of these steps:

    1. Detection of plate regions - The first step, which is also the most processing intensive step, involves the detection of regions of interest where potential license plates might exist. A detector based on Local Binary Patterns (LBP), similar to the face detector of [179], is used for this purpose. An image may contain one or more regions of interest, each of which goes though the next phases.
    2. Monochromatic conversion - This part of the sequence is associated with the conversion of the regions of interest detected in the previous phase into black and white images based on algorithms proposed by Wolf-Jolion [180] and Sauvola [181].
    3. Character Analysis - Once converted to black and white, character analysis algorithms take over trying to find character sized regions in ascending order of sizes, by starting to look for smaller ones first. Regions with connected blobs that are roughly similar to license plate characters and equal vertical alignment are marked for processing.
    4. Edge detection - A Hough transformation [182] is employed to detect all four straight edges of the plates. This step also takes into account information like character height from the previous phase, and ratio of actual plate width and height depending on the geographical region to make the best guess of the precise position of the plate in the image.
    5. De-skewing - Now, any rotation or skew that might exist is corrected by remapping the plate region.
    6. Segmentation of characters - This phase is related to cleaning of the plate region by removing speckles and edges, so that they are not mistaken for characters like the “letter l” or “number 1”. Also, it separates all characters and uses vertical histogram to detect gaps in characters.
    7. OCR - Involves the analysis of individual characters and computation of confidence.
    8. Result generation - In the last part of the sequence, the best possible character combination is searched in the context of the known plate pattern that corresponds to the different regions of the world.

- Selection of the relevant plate - We know more than one plate may appear in an image captured by the device. For example, in addition to the vehicle in front, we may also capture vehicles coming from the opposite direction, or vehicles parked along the road. Thus, to eliminate the probability of false alarms, we have devised the *Same Direction Test*. Using this test, we can detect vehicles in motion and travelling in the same direction as the user of the application. This way, we can ignore vehicles that are parked or moving in the opposite direction, and focus solely on the plate of the vehicle in front of us that is moving in the same direction.

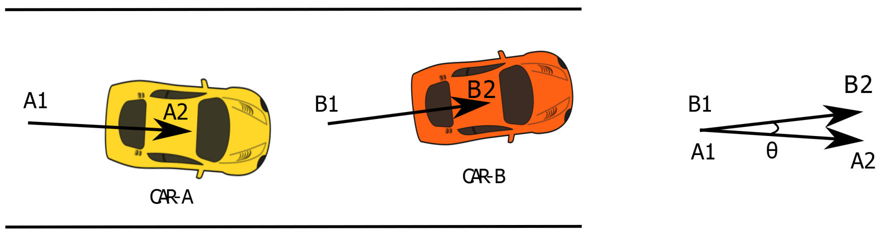


*Figure 3.15: Same Direction Test.*

Figure 3.15 depicts the idea behind the *Same Direction Test*. It involves construction of displacement vectors for each neighbouring vehicle, from the current and previous location information contained in the messages received from them. This information is looked up using the plate number recognised with the help of the smartphone camera, and using the plate number as key for requesting a query to the database storing the messages received. Remember that each vehicle, while broadcasting their location information also sends their actual plate number as the message Id. Now, the displacement vector of the neighbouring vehicle is compared with the displacement vector of the receiving or current vehicle, and the angle between them is measured. If the measured angle is less than a predefined threshold, we consider the two vehicles to be mobile and travelling in the same direction.

Note that only the vehicle ahead will be able to satisfy the *Same Direction Test* when the application is used in scenarios consisting of bidirectional two-lane roads, where only one lane is used by vehicles moving in each direction.

- Distance calculation - For the calculation of distance between two vehicles, we rely on image processing and not on GPS data due to its inaccuracy. It is calculated in the following manner:

  The general lens equation is:

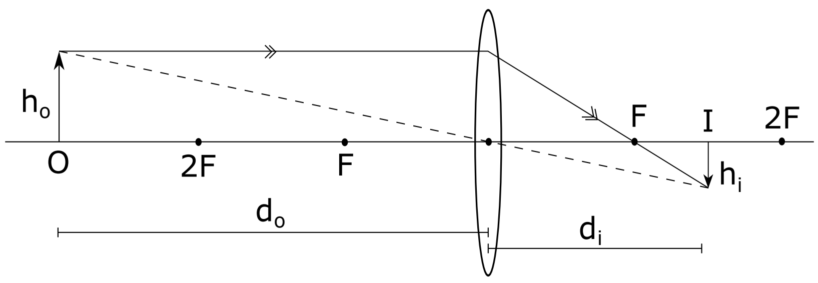


*Figure 3.16: Refraction by convex lens when the object is beyond 2F.*

$$\frac{1}{f} = \frac{1}{d_0} + \frac{1}{d_i} \tag{3.1}$$

where $f$ denotes the focal length of the lens, and it is equal to the distance from the center of the lens to the focal point (F), as can be seen in figure 3.16. Similarly, $d_o$ is distance of the object (O) from the centre of the lens, and $d_i$ is the distance between the lens and the image formed (I).

Also, we know that,

$$\frac{d_o}{d_i} = \frac{h_o}{h_i}$$

$$\therefore d_i = \frac{h_i d_o}{h_o} \tag{3.2}$$

where $h_o$ is the actual height of the object (O), and $h_i$ is the height of the image (I) formed. Since a captured digital photograph may consist of a view in which we have one or more objects of interest, and to avoid confusion, we will refer to I as the *object of interest in the image* for future references.

Now, substituting equation (3.2) in (3.1), we have

$$\frac{1}{f} = \frac{1}{d_o} + \frac{h_o}{h_i d_o}$$

$$\therefore d_o = f(1 + \frac{h_o}{h_i}) \tag{3.3}$$

Values of $f$ and $h_o$ in (3.3) are known, and $h_i$ representing the height of the object of interest in the image, is calculated in metric units for example, as:

$$h_i = k * h_{ipx} \tag{3.4}$$

where $h_{ipx}$ is the height of the object of interest in the digital image, measured in pixels. And $k$ is physical size of the pixel in metric units. Note that digital images may be represented in different resolutions. Thus, the pixel size ($k$) for a image of resolution $M$ pixels wide and $N$ pixels high, is computed as:

$$k = \frac{h_s}{N} \tag{3.5}$$

Here in (3.5), $h_s$ represents the camera sensor height. Although pixel size may also be evaluated using the width of the camera sensor and the digital image. However, for our calculations we rely on the height information of the camera sensor for the sake of uniformity, since we began the whole analysis using the height of the actual object, and the height of the object of interest in the image.

Now, replacing the value of $k$ in (3.4) with the established equation (3.5), we have:

$$h_i = \frac{h_s * h_{ipx}}{N} \tag{3.6}$$

Finally, substituting equation (3.6) in (3.3), we get:

$$\therefore d_o = f(1 + \frac{h_o * N}{h_s * h_{ipx}}) \tag{3.7}$$

Thus, we use equation (3.7) for calculating the distance between the two vehicles from images taken from the trailing vehicle.

- Alert user and the car ahead - Finally, once the distance between the two vehicles has been calculated, we check if the cars are too close, and in that case we generate a warning and also caution the driver ahead by sending an alert making use of the GRCBox communication. Now, we have to define a safe distance, which if not met, would cause alerts to be generated. Many governments agencies like the Road Safety Authority (RSA) [183] of Ireland and the Department of Motor Vehicles [184], State of New York, suggest the two-second rule. According to this rule, the minimum safe distance between two vehicles, one following the other, varies according to the velocity of the vehicle behind. It should be at least the distance that the trailing vehicle would cover maintaining the same speed for two seconds. This should provide enough time for the driver of the car behind to react if the car ahead comes to an abrupt stop. Since the application is intended for deployment in urban scenarios, where speed limits usually varies between 40-60 kmph in most countries, resulting in a safe distance varying within 22-33 meters following the two-second rule. Such large distances are never maintained between vehicles in urban traffic, and so it would render the application unproductive, and could be considered more of a nuisance if it starts warning drivers when the distance between two cars are over 20 meters causing unnecessary distractions. More reasonable, in this case, would be to maintain distance of one car length between two vehicles; in other words, the car behind should leave sufficient gap between itself and the vehicle in front, so that another vehicle could fit in that gap. Since most family-sized cars are within 5 meters length, we select this distance as the minimum safe distance in our application and thus, if the distance between two vehicles is less than 5 meters, it starts displaying warnings.

#### 3.2.2.3. Results

We have performed various tests both in scenarios with and without mobility. The experiments in scenarios without mobility were made to first tune some application parameters, and later check the usability of the application in actual scenarios involving mobility.

1. Static Experiments - In the first scenario we studied the application performance when no mobility was involved. Photographs of cars that were not moving were acquired from a parking area, and these images were processed by the Android devices to fine tune our application. However, prior to that goal, we wanted to make sure our methodology used to calculate the distance between the cars based on the plate size, is accurate enough. Thus, we took photographs of a car at a known distance away from the Android camera, and processed them to calculate the distance.

   Table 3.5 lists the results obtained when the camera was placed at a fixed

| Actual distance (m) | Average calculated distance (m) |
|---:|---:|
| 3 | 3.0 |
| 5 | 4.9 |
| 8 | 7.8 |
| 10 | 9.7 |

*Table 3.5: Comparison of actual distance with calculations using equation* (3.7).

distance of 3, 5, 8, and 10 meters from the back license plate of a static vehicle. Values listed in the table under the heading: "Actual distance" denotes the distance between the license plate and the camera, measured manually. Images were captured by the Android camera, and they were later processed to calculate the distance using our algorithm. The results obtained from processing the images, are listed under "Average calculated distance". This test to prove the developed theory, was repeated between 3-5 times per category of distances mentioned in the table, and the mean value of the results is shown in the table. The reason behind the fewer repetitions of this experiment is due to the fact that similar results were obtained each time. It can be noted that as distance increases, the error in the calculated distance also increases, but we are more or less able to get the distance from the images, thus confirming that the developed theory holds. The error involved is nearly insignificant when compared to GPS related errors. The major causes for this error are: optical deformation of the lens, unsharpness of the image being analysed, and the effects of digitisation by the camera. These errors can be minimised by proper calibration [185] of the camera and later compensating for any deformation. But, in our application we have relied on data supplied by the manufacturer for calculations in the attempt to make the software easy to use, without the users being troubled for calibrating the camera.

The processing time of images is an important parameter for determining the usability of our FCW application. Thus, in our initial experiment, we wanted to study the time taken by different Android devices to process and identify the license plates for various resolutions. In this experiment we studied the time taken by five different devices, namely Nexus 7 tablet, Motorola Moto G-3, Nexus 5X, Nexus 6, and Samsung Galaxy Note 10.1. The Nexus 7 had a quad-core 1.2 GHz processor and 1 GB Random Access Memory (RAM). Similarly, the Moto G-3 possessed 1.4 GHz quad-core processor and 2 GB RAM. The Nexus 5X was equipped with a 2 GB RAM, and hexa-core processor with four cores running at 1.4 GHz, and the other two at 1.8 GHz. While the Nexus 6 had a specification of 3 GB RAM and a 2.7 GHz quad-core processor. Finally, Note 10.1 came with a 3 GB RAM and 2.3 GHz quad-core processor.

Figure 3.17 shows the time taken to process and identify plates in images of High Definition (HD), Video Graphics Array (VGA) and Quarter Video Graphics Array (QVGA) resolutions for the different devices. The time taken for processing HD images varied from 1.8 to 4.2 seconds, depending on the device, while it ranged from 1.4 to 3.3 seconds when considering VGA, and for QVGA it was between 1.1 to 2.6 seconds. Thus, lower resolution images were processed faster, and devices with faster processors

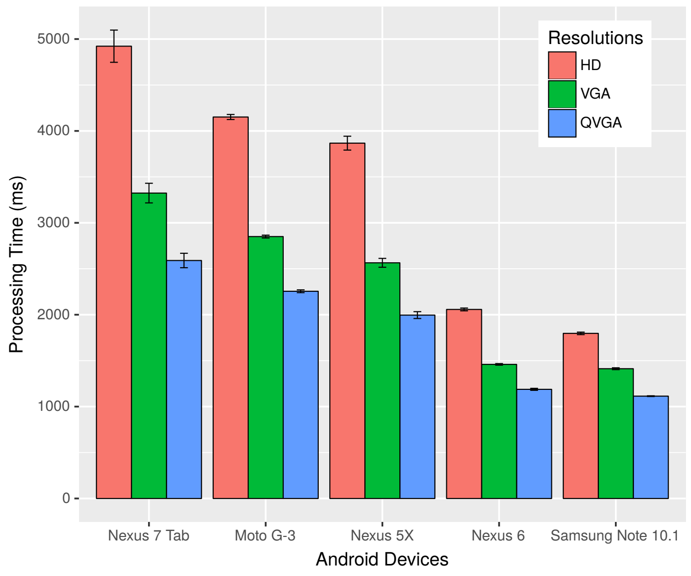


*Figure 3.17: Time taken to process images of different resolutions by Android devices.*

performed better. The Samsung Note shows the best performance, followed by Nexus 6, Nexus 5X, Moto G-3, and Nexus 7 tablet. Note that, even with the best processing times achieved by the Samsung Note of 1.8 seconds for HD, 1.4 seconds for VGA, and 1.1 seconds for QVGA resolutions, this time overhead is still very high when compared with dedicated image processing devices.

The Android OS, while encoding JPEG images, accepts a parameter called quality whose value can range between 0 to 100. The value of 0 produces images of maximum compression, while 100 compresses for maximum quality. Next, we wanted to check if this parameter has a role to play in the processing time of images for plate recognition.

In figure 3.18, we can see that, when the value of the quality parameter is varied from 20 to 80, the processing time increased sightly for higher values of quality. Higher resolution images are more affected than the lower resolution ones. We have varied the value of the quality parameter from 20 to 80 since values below 20 resulted in images with very low visually perceived quality, while values over 80 did not produce any significant improvement. All the processing was done by one device, which was the Moto G-3 in this case. Note that, for the QVGA resolution, the processing time is nearly fixed at 2.2 seconds even with the variation of the JPEG quality, while for VGA it rises from 2.7 to 2.8 seconds, and for HD images it ranged from 3.7 to 4

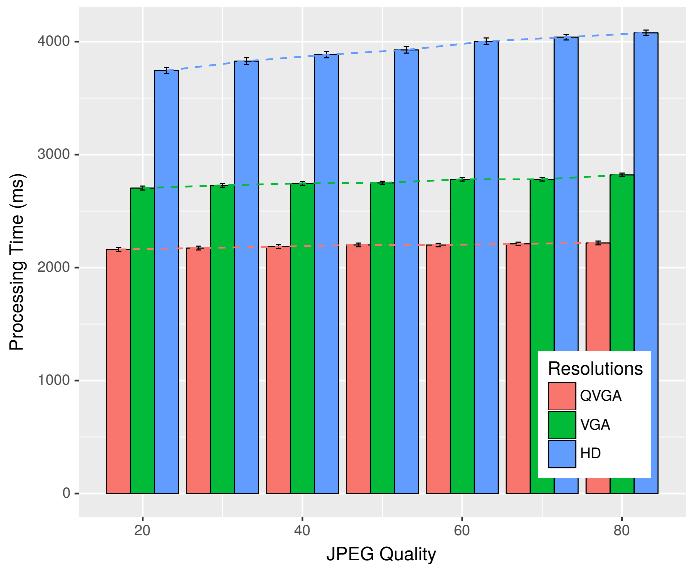


*Figure 3.18: Plate processing time with Moto-G3 for different JPEG qualities.*

seconds.

So far we have seen that lower resolution images are processed faster, and that image quality has very little or no effect on the processing time for lower resolution images. While, in case of higher resolution, there is an increase in the time taken to detect plates. Another important factor that also should be taken into account is the accuracy or the degree to which the identified plate number and the actual license registration number match. Also, it is important to check whether parameters like image resolution and quality has any effect on the accuracy of the identified plate.

Figure 3.19 depicts the effects of JPEG image quality on the accuracy of plate recognition. An accuracy of 1 reflects an exact match where the identified plate perfectly coincides with the actual plate, while 0 indicates no match or no plate was detected in the image; values between 0 and 1 imply a partial match. Note that the resolution of QVGA shows a huge improvement of accuracy, varying from 0.12 to 0.5, with the boost in the image quality. For higher resolutions of VGA and HD, it ranged from 0.83 to 0.89, and lied between 0.89 to 0.91, respectively, depending on the quality parameter. This suggests the use of higher resolutions rather than lower ones, for the purpose of our application.

Another physical factor that could affect the performance of our application is ambient light. Thus, to study the effect of lighting conditions, we decided

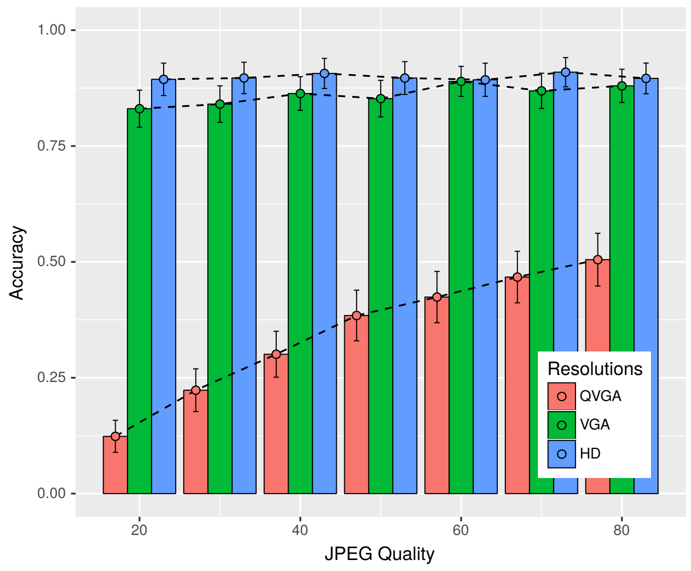


*Figure 3.19: Accuracy of plate recognition for different JPEG qualities.*

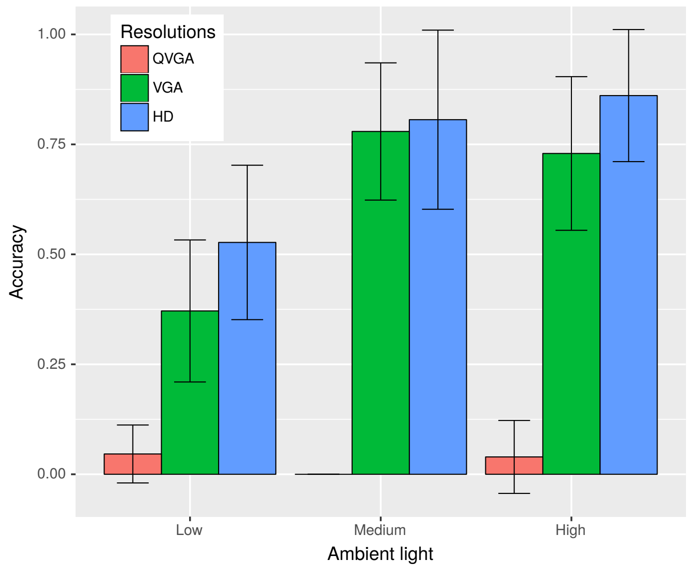


*Figure 3.20: Accuracy of plate recognition under different lighting conditions.*

to perform three sets of experiments under various lighting conditions at daytime, dawn/dusk, and during the night.

Figure 3.20 illustrates how the accuracy of the plate recognition changes with altered lighting conditions for various resolutions. Here, low lighting conditions refers to conditions during the night with the presence of insufficient light from street lamps to nearly no light at all in some images, resulting in grainy, unclear and dark images. Medium ambient light is referred to the time when the sun is just about to rise or right after it has set. Lastly, high ambient light encompasses all tests that we have performed during daytime. From the graph we can see that, for QVGA resolution, in this set of experiments we had very little success recognising plates, even during daytime. For VGA images, accuracy varied from 0.37 to 0.78, with best performance in medium lighting conditions, as reflections from the plates caused problems in recognition when exposed to high ambient lighting. For the highest resolution of HD, we were able to achieve accuracy values between 0.52 and 0.86, depending on the amount of ambient light. Note that, due to the small size of the dataset used for this experiment, the confidence interval was high in all the cases.

Thus taking into account all that we have learnt so far from our experiments, we decided to use the HD resolution due to its dominant performance in different lighting conditions and better accuracy of recognition compared to the lower resolutions. Regarding which quality settings to use for the HD resolution, we decided on the quality value of 70, even though, in our experiments in static scenarios, we found that the image quality had little effect over accuracy in case of HD resolutions, and it also resulted in the increase of processing time when using higher quality images. However, taking into account that real scenarios would involve motion, blurring, and effects of vibrations, which would make it harder to recognise the plates, we consider that quality could have a role to play. Also, from figure 3.19 and figure 3.18, we had the best accuracy (0.91 or 91%) for HD at a quality value of 70, but of course the processing time with Moto G-3 was 4 seconds on average. This is about 8% more than the time required to process HD images of the lowest considered quality settings of 20. Thus, the default settings use JPEG images of HD resolution with a quality value of 70, which the user can change according to his or her needs.

2. Dynamic Experiments - In this section we present the results we have achieved in the tests we performed with our Android FCW application, using the settings we have defined from the study of the application in scenarios without mobility. Experiments performed in this section were undertaken using two real vehicles, one following the other at all times, each equipped with a GR-

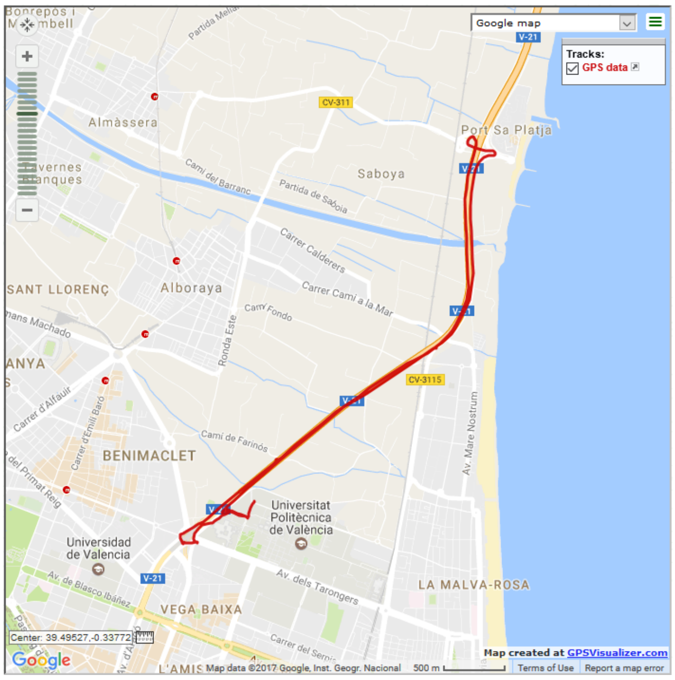

Google map
Tracks:
GPS data
Almàssera
Alboraya
Saboya
Port Sa Platja
BENIMACLET
Universitat Politècnica de València
Universidad de Valencia
VEGA BAIXA
LA MALVA-ROSA
Center: 39.49527,-0.33772
Map created at GPSVisualizer.com
Map data ©2017 Google, Inst. Geogr. Nacional 500 m
Terms of Use Report a map error

Figure 3.21: Route-I without many curves.

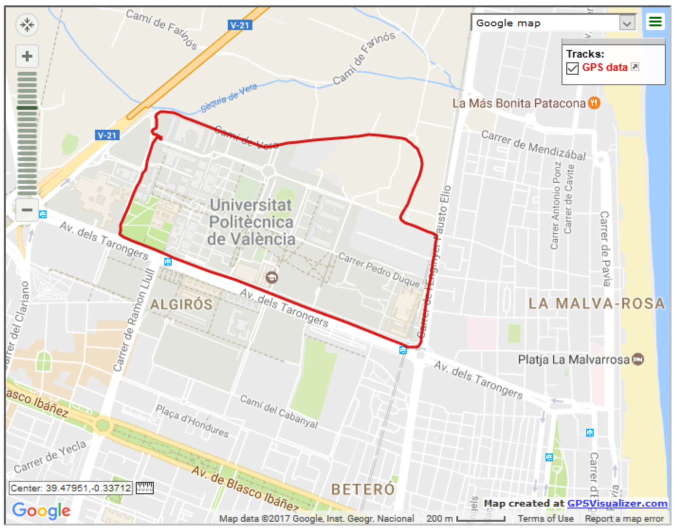

Google map
Tracks:
GPS data
La Más Bonita Patacona
Universitat Politècnica de València
ALGIRÓS
LA MALVA-ROSA
Platja La Malvarrosa
BETERÓ
Av. dels Tarongers
Center: 39.47951,-0.33712
Map created at GPSVisualizer.com
Map data ©2017 Google, Inst. Geogr. Nacional 200 m
Terms of Use Report a map error

Figure 3.22: Route-II with some curves and turns.

CBox for communication and an Android tablet running the developed application, as shown in figure 3.13. The aim of our outdoor experiments with mobility was to try and find a good threshold value for the *Same Direction Test* that is used by the application to discard plates of vehicles not in motion or coming from the opposite direction, and to see how our application performs in challenging real scenarios.

Figure 3.21 shows one of two routes taken during our outdoor experiments. This route was about 9.25 km long with very little turns and curves. On the other hand, figure 3.22 depicts a 3.76 km route, around the Universidad Politècnica de València, with some turns and curves.

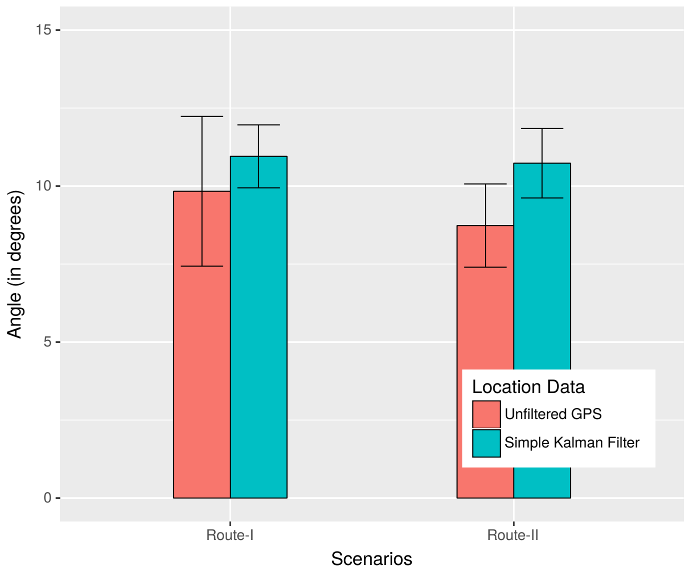


*Figure 3.23: Results of the* Same Direction Test*.*

Figure 3.23 presents the results from the *same direction test*, that is used to detect if vehicles are travelling in the same direction. A comparison has been made between the use of unfiltered GPS locations and Kalman filtered [186] location data for the evaluation of the *same direction test*. The Kalman filter used here is a simple one that just takes into account the location data. From this graph we can see that the average angle evaluated by the *same direction test* using unfiltered data for scenarios presented in figure 3.21 and figure 3.22 are 9.83 and 8.73 degrees; instead, when using the Kalman filter, similar values of 10.95 and 10.73 are observed, respectively. Notice that, for both filtered and unfiltered data, the worst case values are within 12.5 degrees. Thus, 12.5 degrees is selected as the default threshold for the *same*

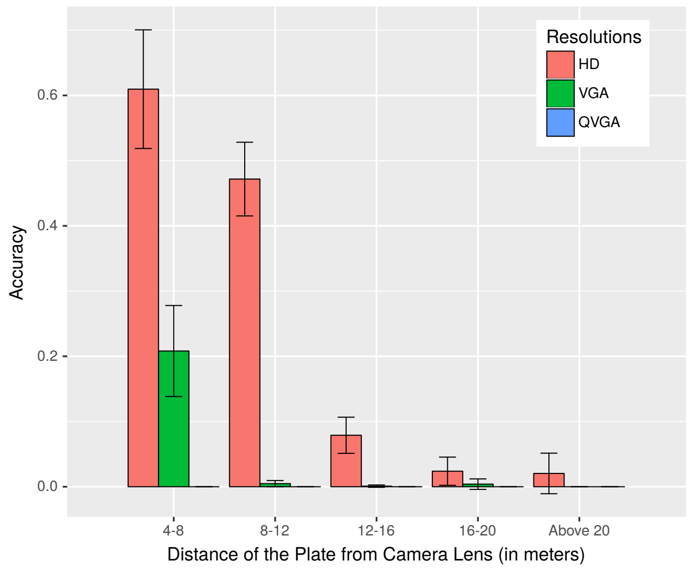


*Figure 3.24: Accuracy of the recognised plates for varying distances during daytime, in scenarios involving motion.*

*direction test* in our application.

During our assessment of the *Same Direction Test* involving real cars, we also studied the effect of distance on plate recognition as we tried to identify the license plate of the car ahead. Figure 3.24 displays the results obtained. This graph reassures us of our choice of HD resolution as the default image resolution for our application, as other lower resolutions fail to perform well in this scenario. Let us look closely at the first two groups of observation taken between 4-8 meters and 8-12 meters; as the car behind comes closer to the vehicle ahead, the accuracy of the recognised plate increases. Between 4-8 meters the accuracy is about 0.61 for HD images, which is much lower than what we observed in our experiments without mobility. The lower accuracy is the result of increased number of failures when attempting to recognise the plate as a consequence of the problems that the device faced to stabilise the images due to vibrations resulting from motion. Note that, in our experiments, we have also included results when two vehicles were 4 meters apart, even though warning generation starts at 5 meters, because the application needs to keep alerting the drivers even when the distance is less than 5 meters. We have no results from below 4 meters as it was difficult to emulate such dangerous situations in real experiments and also because sometimes the plate of the car ahead was outside the captured frame when the two cars were very close to each other.

Finally, we also tried to repeat the same experiments involving motion, during night time, to make a comparison of how the application performs under low lighting conditions. We were unable to identify plates during the night using all three different resolutions due to several reasons: the camera equipped in the smartphones found it hard to focus under low lighting conditions; vehicles usually have lights near the license plate that are present to help illuminate the plate in darkness, but these lights, in our case, made it more difficult for the camera to focus; and, when the car behind closed in on the car ahead, reflections due to its headlights made the plate illegible.

#### 3.2.2.4. Conclusion: Forward Collision Warning

We have presented an Android FCW application that uses plate recognition and inter-vehicular communication to alerts drivers on getting too close to the vehicle in front. We have performed various experiments with our application and found out that, although mobile devices with faster processors tends to function better, high end models still take too long to recognise license plates (in the order of seconds), which is the strongest reason impeding the adoption of the solution. Critical safety applications, such as the one we have presented here, need to process more than one image per second, for performance-related reliability. Also, the camera on the devices was not powerful enough to stabilise images captured when in motion and conditions involving low light. Thus, we might still have to wait for more powerful devices of the future. Apart from these hardware-related issues, our application is able to function in bidirectional two-lane roads, capable of detecting plates of cars coming from the opposite direction or static parked cars, and discard these cases without warning generation with the help of a test that we have designed, named the *Same Direction Test*. Even though this test performs well, it does not take into account multiple lane roads where cars would be travelling in the same direction. Thus, situations might arise where unwanted warnings are generated upon the identification of the plate of a vehicle travelling on a different lane but in the same direction, which is within the communication range of the vehicle using our application. This problem has been left as future work and needs to be addressed.

# 4 EYES: The Video Overtaking Aid

The ITS area has experienced great developments in the recent past, although suffering from slow adoption rates, thus depriving consumers of many interesting and innovative applications. The only way to resolve this problem is to develop ITS solutions using the already available technologies that are within the grasp of the common people, to make them cost-effective, quick to deploy, and easy to adopt. We have therefore developed an affordable ITS system that makes use of standard smartphones to assist drivers when overtaking.

Accidents while overtaking are considered by many sources [187] as one of the main reasons behind injuries and loss of lives on the road. Owing to the fact that scarce opportunities arise to practise overtaking during standard driving lessons, it is no surprise that errors may be committed by both inexperienced and experienced drivers alike. Groeger and Clegg [188], in their analysis of manoeuvres in lessons that stretched over 550 hours, deduced that practising overtaking only formed 5% of the total duration. Once identified as a major and critical problem, overtaking was studied in detail at the University of Nottingham, and their findings can be accessed at [189]. Thus, our aim was to address this problem and make the roads safer by developing a real-time visual overtaking assistant application that works without user intervention.

The application developed aims at providing visual overtaking assistance, and it runs on the Android platform. The system autonomously creates a network among close-by vehicles, and it provides drivers with a real-time video feed from the one located just ahead. Our system seamlessly offers a better view of the road, and of any vehicle travelling in the opposite direction, being especially useful when the front view of the driver is blocked by large vehicles. We have validated our overtaking assistance system in both laboratory environment and realistic scenarios. The minimum hardware requirement for our application is the use of a smartphone equipped with GPS, wifi, and a back camera. The smartphone running our application is to be mounted on the vehicle windshield, and the camera is used to record a video which is transmitted over the vehicular network to the vehicle located just behind where it is displayed. This way, it provides an enhanced multimedia information aid to the drivers based on which they might decide whether to overtake. It is important that the devices being used possess GPS because, based on location information, both the source and destination of the video stream are chosen. It is to be kept in mind that the video streaming occurs between cars travelling in the same direction, and always occurs from the vehicle in front to the vehicle travelling behind. The smartphone is to be mounted in such a manner that its screen faces the driver, and the back camera points towards the windshield. Care should be taken that the camera has a clear view of the road in front, and of the cars coming from the opposite direction, so that, when the video is streamed, the driver of the car behind is made aware of the traffic situation ahead of it. The drivers would only receive and check this video when they wish to overtake the vehicle ahead, basing their decision on what they see in the video, being especially useful in scenarios where the view of the driver is blocked by a larger vehicle, or when a long queue of cars is located ahead and the driver wishes to overtake. It is important to note that our application works without user interaction once started. Also, the video streaming and playback always occurs between the car just in-front and the vehicle following it to eliminate any confusion that might arise if the video was streamed by the leader of the queue. In such a case, the driver, unaware of the number of cars ahead, would be overtaking in dangerous situations. Another added advantage of using this type of communication is that our application does not suffer from typical multi-hop delays.

We have evaluated the developed application in both indoor and outdoor scenarios. The indoor tests involved comparing the performance of the application using two different video codecs, namely H.264 [190] and MJPEG [1], which involves compressing the video stream separately as JPEG [191] images. These two encoding formats were compared focusing mainly on the delay between capture and playback of the video stream. Then, choosing the best codec based on the indoor experiments, we have performed outdoor tests involving real cars. A more detailed explanation about the developed application in terms of its architecture, design, implementation issues, and results obtained will be provided in the following sections.

[1]More on MJPEG: http://en.wikipedia.org/wiki/Motion_JPEG

## 4.1. Application Overview

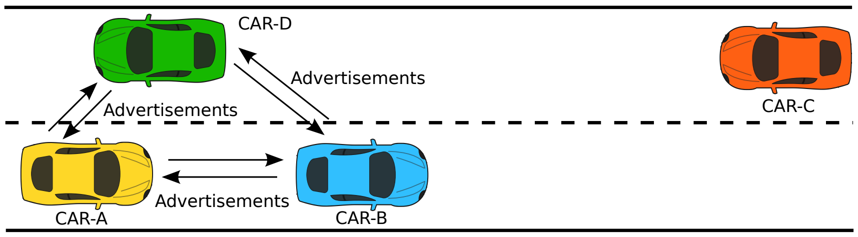
Our application enables vehicles to perform real-time streaming of the view as seen by the driver sitting in the car ahead, thus providing users with visual assistance during overtaking. This is specially useful in bidirectional two-lane highway scenarios with one lane allotted per direction of traffic movement. For our application to function properly, the users need to have a smartphone with GPS and a back camera, besides the availability of the vehicular network for data transmission.

The working of our overtaking aid application can be explained in three easy steps. In *step one* the sender and the receiver vehicle of the video stream is selected using some special tests and conditions. The *second step* involves the transmission of the video being captured by the sender, and its display at the receiver end. In the *third step*, which is also the last one, video transmission and playback are stopped when video transmission is no longer necessary.

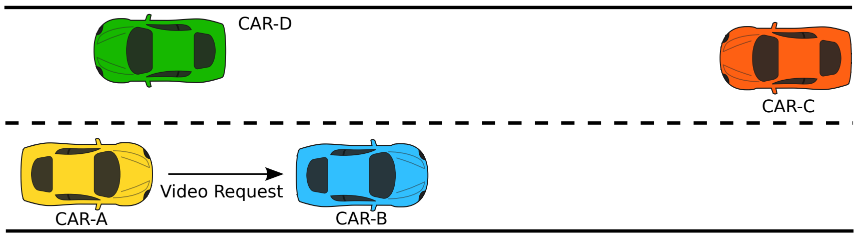
*Step one*, which involves the election of video source and destination, begins with the broadcast of an advertisement message by devices running our application. This advertisement message is basically an announcement of video availability by nearby vehicles, containing information regarding the location of the vehicle broadcasting the advertisement packet, and its direction of motion. Each vehicle, while broadcasting the video availability message, also listens for advertisement messages coming from its neighbours. Upon receiving an advertisement packet, the vehicle receiving it verifies whether the sender is a valid source from which a video stream may be requested. This *validity check* involves tests to find out whether the source is travelling just ahead of the receiver on the same lane, and in the same direction. If more than one valid video source is detected, then the receiver selects the best source from all valid sources requesting the video. The selection is based on the distance between source and receiver. A more detailed explanation of the *validity check* to select the video source is provided later in the section to follow.

The selected source vehicle, upon receiving the request for the video, starts streaming the video signal to the destination over the vehicular network in *step two*. However, before starting the video transmission, the source checks the validity of the request for video by performing the same *validity check*, used by the destination node before sending the request in *step one*. At the destination, the video is displayed on-screen for the driver as soon as its reception starts. The streaming of the video by the sender, and its playback at the receiver end, is stopped when the receiving vehicle successfully overtakes the sender, or when it stops following the sender; in either case the video stream becomes irrelevant.

Figure 4.1 depicts *step one* previously explained. In the case depicted in this figure, there are four cars, and all of them are using our application. CAR-A and



(a) The vehicles exchange *advertisements*.



(b) The client *requests* video from the server.

*Figure 4.1: Functional overview of the application - Step one.*

CAR-B are moving in the same direction, while CAR-C and CAR-D are moving in the opposite direction. In the beginning, all the cars broadcast advertisement packets containing information about the sender location and the direction of motion, as shown in figure 4.1a. CAR-C is outside the communications range of all the other cars, and hence it is unable to receive any packets transmitted by them. Upon receiving advertisement packets, each car performs the *validity check* to detect if the source of the advertisement is travelling ahead of it, on the same lane and direction. Here, only CAR-A finds the advertisement message sent by CAR-B to be valid, and thus sends a video request to CAR-B, as depicted in figure 4.1b.

Once CAR-B receives the video request, it performs the *validity check* to ascertain if the sender of the video request is following it, and travelling on the same lane. Since here the *validity check* is satisfied, CAR-B starts streaming video to

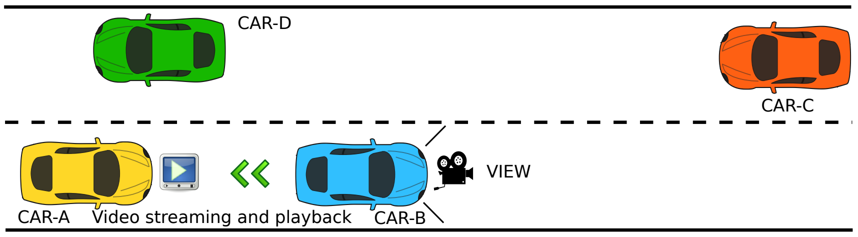


*Figure 4.2: Functional overview of the application - Step two.*

CAR-A, as shown in figure 4.2. CAR-A then starts playing the video stream for its driver as soon it starts receiving the stream. Notice that a particular vehicle may act as both the video source and destination, a situation which might arise when there is a queue of cars travelling in the same direction or platooning. In that case, a vehicle might receive a video aid from the node travelling ahead, while streaming its own view to the node following it.

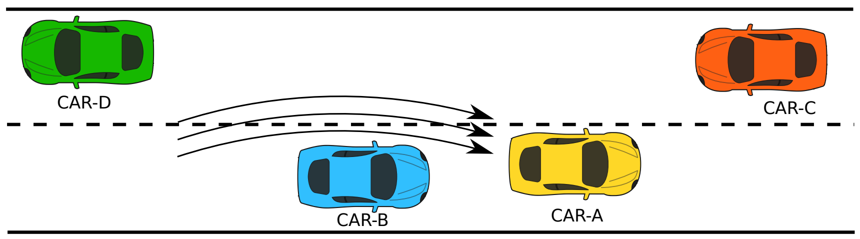


*Figure 4.3: Functional overview of the application - Step three.*

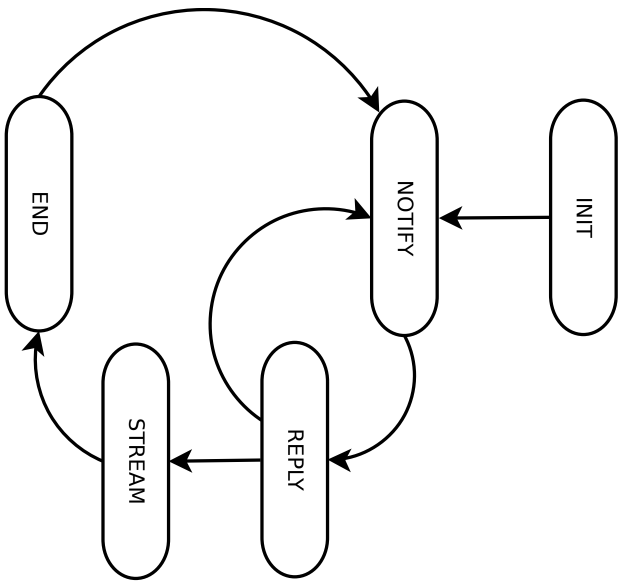
Figure 4.3 points out one of the two cases which might lead to the end of video streaming and playback. In the figure, we can see that CAR-A has overtaken CAR-B, which causes the video streaming to stop. Now, since CAR-A is travelling just ahead of CAR-B and in the same direction, CAR-B might request overtaking aid from CAR-A.

## 4.2. Implementation Details

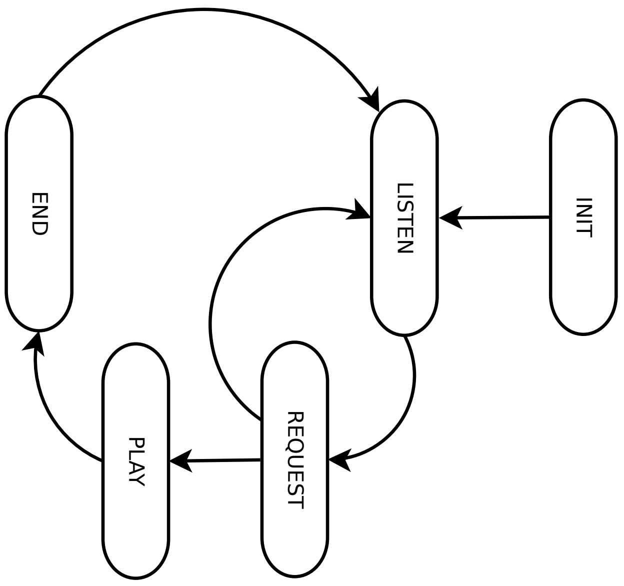
In the previous section we have seen an overview of the main application features. In this section we are going to see in detail the different working components of our proposed application, and how they work together to provide users with the visual aid while performing overtaking manoeuvres.

### 4.2.1. The Video Server and Client

Apart from the fact that the functionality of our application can be split into three steps, we are also aware that each node running our architecture can work as a server and client at the same time. This means that all nodes have the capacity to receive video from a node and display it, while streaming video to a completely different node. However, for the sake of clarity, we are going to consider two devices running our application placed in two different cars, out of which only one will be acting as the server streaming video, while the other will act as a client by merely receiving the stream. Even though at the beginning of *step one* the roles of vehicles are not defined, we will refer to vehicles as server and client according to the roles they will be attaining in the future.



(a) Different Server States.



(b) Different Client States.

*Figure 4.4: State diagram of the Server and Client.*

Figure 4.4 presents all the possible states of the client and server. At the outset of *step one*, the server is in the *notify* state, while the client is in the *listen* state, as displayed in figures 4.4a and 4.4b. The server, in the *notify* state, starts advertising the availability of the video feed by broadcasting an advertisement message (*hello* packet), and also listens for replies from clients. The *hello* message accommodates location and direction information of the server, so that the client, upon receiving the message, can extract the location information and use it to analyse whether the server is travelling ahead and in the same direction. The client, on the other hand, remains listening for advertisements from servers, being in the *listen* state from the beginning, as depicted in figure 4.4b. Upon receiving *hello* messages from servers, the client performs the *validity check* and then, if the test conditions are satisfied, the information is stored in a queue of candidate servers.

The *validity check* used to initiate video streaming consists of three test conditions, namely the *same direction test*, *ahead test*, and the *same lane test*. The *same direction test*, as the name suggests, is used to detect whether the two vehicles are travelling in the same direction. With the help of the *ahead test*, the application makes sure that the source of the video stream is travelling ahead of the receiver. Once the *same direction test* and *ahead test* have been fulfilled, the *same lane test* is used to be sure that the leading vehicle and the one behind, are travelling on the same lane. We will take a look at these tests in detail later in this chapter. Hence, if the *validity check* is satisfied, then the two vehicles are assumed to be travelling in the same direction, one following the other on the same lane, and the vehicle behind may request the video stream from the vehicle ahead.

The client, which was in the *listen* state, after having prepared a list of all valid servers performing the *validity check*, selects the best server based on the distance between the server and client. Next, the chosen server is sent a *request* for the video stream by the client, which moves to the *request* state. The server, upon receiving the *request* from the client, executes the *validity check* before replying to the client with a *ready* or *reject* message. A *ready* message is sent if the outcome of the *validity check* is positive; otherwise, a *reject* is transmitted, and the server state changes to *reply*. The server state may further change to *stream*, or move back to the *notify* state, depending on its own reply to the client. On the other hand, the client state changes from *request* to *play* only if the reply from the server was a *ready* message containing the *video sender port* number; otherwise, it goes back to the *listen* state. Table 4.1 lists the different packets exchanged between the server and the client.

*Step two*, which involves video streaming and playback at the server and client ends, respectively, begins only if the server is in the *stream* state, and the client is in the *play* state. Apart from streaming video, the server in this step sends a *data* message containing its location, direction, and speed information to the client. Such *data* message is relayed every second, and the purpose of this message is to

| Message Type | Message Source | Client State | Server State | Message Contents |
|---|---|---|---|---|
| Hello | Server | Listen | Notify | Location and Direction |
| Request | Client | Request | Notify | Location and Direction |
| Ready | Server | Request | Reply | Video sender port |
| Reject | Server | Request | Reply | - |
| Data | Server | Play | Stream | Location, Direction and Speed |
| Data-Ack | Client | Play | Stream | - |
| End | Client | Play | Stream | - |

*Table 4.1: Messages exchanged between the Server and Client.*

detect if video streaming is necessary. The client, upon receipt of the *data* message from the server, verifies if the conditions of the *validity check* are fulfilled. If not, it is assumed that the video streaming is no longer necessary. If the *validity check* returns a positive result, it responds to the server by sending a *data-ack* message to keep the video connection alive.

If video streaming is no longer necessary, which would be the case on completion of overtaking manoeuvres, or if the car behind stops following the car ahead, *step three* is initiated. In this step, the client requests the server to stop the video streaming by sending an *end* message. The client state changes to the *end* state before switching to the *listen* state, once again during *step three*. On the other hand, the server may end video streaming upon receipt of the *end* message from the client, or if no *data-ack* message is received (it is used to keep the connection between the server and client alive). In our implementation, we have fixed the waiting time for a *data-ack* to 3 seconds, because the selected waiting time is considered to be adequate enough as all communication occurs between two cars, one just ahead of the other.

## 4.2.2. Validity Check

Includes a group of conditions imposed by the application to detect which vehicle should stream video, and which one should receive it. It consists of three test conditions called *Same Direction Test*, *Ahead Test*, and *Same Lane Test*. As the name suggests, the *Same Direction Test* is used to find out if the two cars are travelling in the same direction. The *Ahead Test* is a condition to evaluate if the source of the video is ahead of the video destination. Lastly, the *Same Lane Test* is used to asses if both the vehicles are travelling on the same lane.

### 4.2.2.1. Same Direction Test

This test is used to find out if two cars are travelling in the same direction, and is similar to the one used in our Android FCW application. The only difference is that this test used in our overtaking application has been modified to use multiple GPS observations for construction of the displacement vector of the vehicles.

As can be seen from figure 4.5, we construct displacement vectors of the cars travelling ahead and behind. These displacement vectors are constructed using multiple GPS points, and later regression [192] methods are employed to improve the accuracy. We then calculate the angle between the two vectors. If the calculated angle is under 12.5 degrees, then we consider that the cars are travelling in the same direction. The threshold of 12.5 degrees was established from previous experiments with the FCW application.

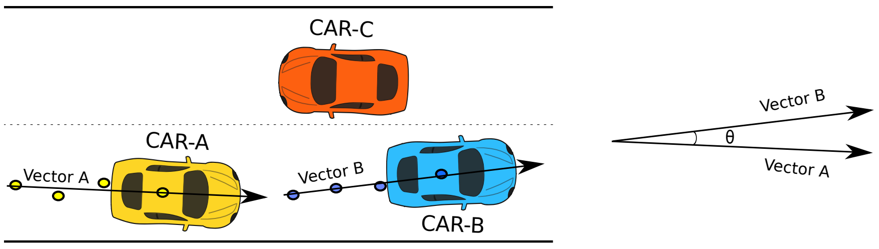


*Figure 4.5: Same Direction Test.*

#### 4.2.2.2. Ahead Test

This test is used to evaluate if one of the vehicles is travelling ahead of the other. The importance of this test lies in the fact that the video streaming in our application always takes place between the car ahead and the vehicle behind, when both of them are travelling in the same direction and on the same lane.

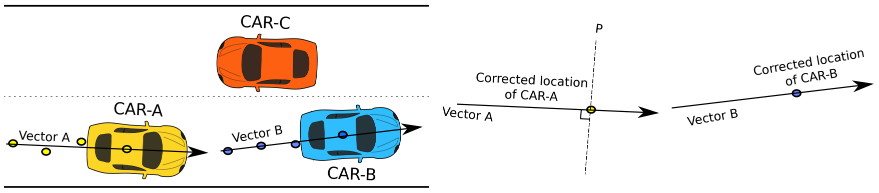


*Figure 4.6: Ahead Test.*

Figure 4.6 shows how the *Ahead Test* is performed. We construct displacement vectors of the two vehicles using its current and past locations. Then, using regression analysis, we try to correct the lateral error in their location. We then construct a perpendicular line (P) at the current corrected location of the vehicle travelling behind. Now if the current corrected location of CAR-B lies to the right of line P in this case, then we can safely assume that CAR-B is ahead of the other car.

#### 4.2.2.3. Same Lane Test

This test evaluates if the cars are travelling on the same lane, and it is used to make sure that video streaming does not start from the vehicle behind to the vehicle ahead unless it has completed the overtaking manoeuvre, and it assumes a new position in front of the vehicle that was travelling ahead of it previously.

Our video based overtaking aid is to be used in bidirectional two-lane highways, where one lane is used for either direction of traffic; notice that in other cases, like one-way streets or roads with multiple lanes for same direction of traffic, it would not be useful. On first thought, it would seem that the two tests: *Same Direction Test* and *Ahead Test*, would be sufficient to detect which car is travelling

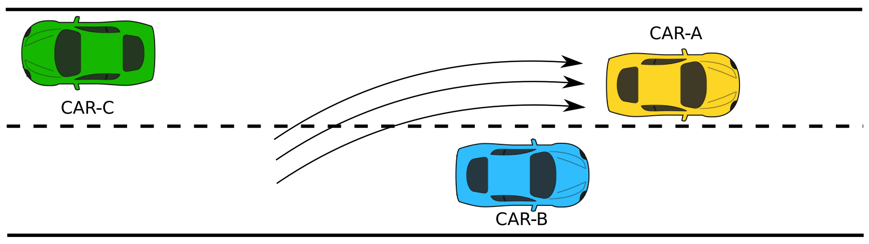


*Figure 4.7: Importance of the Same Lane Test.*

ahead, and to distinguish them from vehicles coming from the opposite direction. However, figure 4.7 shows why a third condition is important. In this figure we see that CAR-A and CAR-B are moving in the same direction, while CAR-C is travelling in the opposite direction. CAR-A, that was previously following CAR-B, starts receiving video from it, and decides to perform an overtaking manoeuvre. While CAR-A tries to overtake, it will eventually reach a point where it is on the other lane, ahead of CAR-B, and travelling in the same direction. At this point, if we apply just the *Same Direction Test* and *Ahead Test*, video streaming would start from CAR-A to CAR-B, while it is on the other lane. Thus, to avoid this type of situations, the *Same Lane Test* plays an important role.

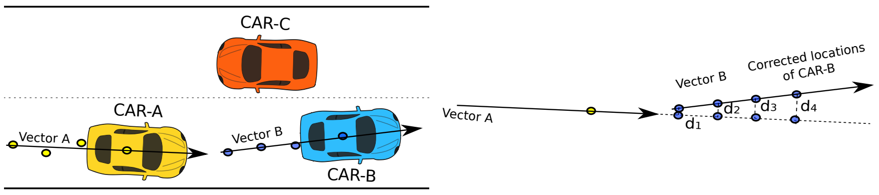


*Figure 4.8: Same Lane Test.*

On looking closely at figure 4.8, we observe that the Same Lane Test involves extending the displacement vector of the car travelling behind, and finding the projection of all the corrected locations of the car ahead, on the displacement of the car behind. We then find out the distances $(d_1, d_2, d_3, ...)$ between the corrected positions of the car ahead and their projection on the vector of the car behind. The average of these distances $(d)$ is used for the evaluation of the test. If this distance $(d)$ is less than equal to half the width of the lane, then the two cars are on the same lane. Here we do not just compare the distance between the current corrected location of the vehicle ahead and its projection on the displacement vector of the vehicle behind, because in our preliminary trials with the *Same Lane Test*, we observed that the use of the average distance improved the reliability of this test.

For improving the accuracy of the test, we take advantage of the concept that the EYES application is only useful in undivided dual carriageways where there is traffic in both directions, and traffic in each direction uses a single lane. Thus, we can make the assumption that for right hand traffic, if the corrected location of the car ahead lies to the right of the displacement vector of the car behind, then no matter the value of $d$, we can assume the cars are on the same lane.

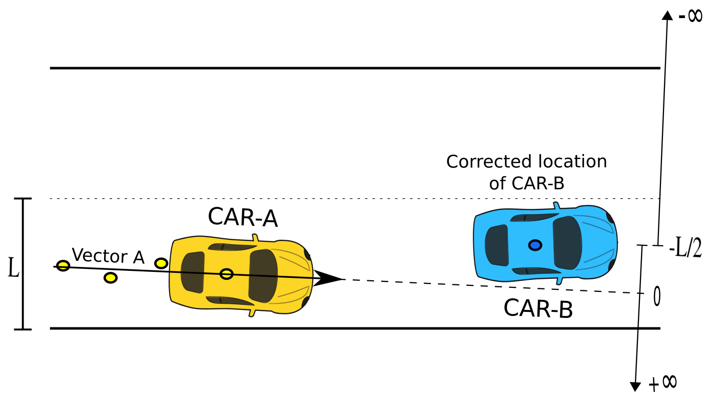


*Figure 4.9: Limits of the* Same Lane Test.

Figure 4.9 better explains the idea. The displacement vector of the car behind is considered to coincide with the centerline of the lane. Thus, if the car ahead is currently anywhere from $+\infty$ to $-L/2$ from the displacement vector of the car behind, then we consider that the cars are on the same lane. Note that here $L$ represents the width of the lane.

Next, we want to make a theoretical analysis of the developed *Same Lane Test* to evaluate the kind of performance we could expect from it. Here we assume that we are using two GPS positions to construct the displacement vectors of the vehicles, and that the error involved in the position can be characterised by a normal distribution, to simplify the interpretation of the test.

For the purpose of the analysis, let us consult figure 4.10. Assume that the current and previous position of the vehicle behind (CAR-A) is known through

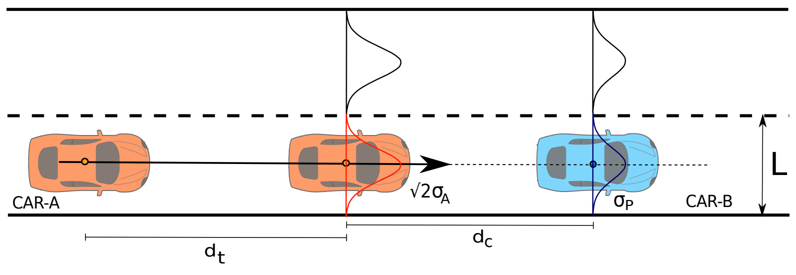


*Figure 4.10: Theoretical analysis of the* Same Lane Test.

differential GPS up to an immediate error characterised by a normal distribution with standard deviation $\sigma_A$ . Given two measurements of its position separated by a distance $d_t$, the uncertainty on the direction of its driving vector is given by $arctan(\sqrt{2}\sigma_A/d_t)$, assuming that the uncertainly in both their positions will be similar since they are close to one another, and receive location update from the same satellite constellation. Under the hypothesis that the driving vector is centered and parallel with respect to its driving lane, the uncertainty distribution of the lateral position of the centerline of the lane at a distance $d_c$ in front of the vehicle has standard deviation $\sigma_P = \sqrt{2}\sigma_A(d_c + d_t)/d_t$. The same uncertainty distribution applies to the position of the adjacent driving lane, with its mean shifted by the lane width $L$. The True Positive Rate (TP) of the *Same Lane Test* is the area of the own lane's normal distribution right of the lane separation, and is equal to the accuracy of the test, calculated as follows:

$$Accuracy = TP = \frac{1}{2} + \frac{1}{2}erf(\frac{d_t L}{4(d_c + d_t)\sigma_A}) \tag{4.1}$$

### 4.2.3. Video Transmission

Our application relies on the Real Time Streaming Protocol (RTSP) [193] for sending video over the vehicular network. It is used to establish and control the media sessions between the server and the client while they are in the *stream* and *play* states, respectively.

Figure 4.11 illustrates all the data and commands exchanged between the server and client. The server (video sender) initially listens for any incoming connection from the client (video receiver) on a predefined port, here we assume it is port A. This port is made known to the client with the help of the *ready* message sent by the server in response to the video request made by the client. Thus, the client communicates with server at port A, making use of local port X to send or receive packets.

The entire communication between the server and client can be explained in three steps:

1. Step-I: used to setup the video streaming process.
2. Step-II: the video data transfer takes place.
3. Step-III: includes the exchange of data to terminate the video streaming.

In Step-I, the client enquires all the *options* supported by the RTSP server. The server replies to the request by listing all the commands that it supports, which in our case includes *describe, setup, play* and *teardown*. The client, by using the *describe* command, asks the server about the video properties that the server would be sending, to which the server replies. Afterwards, the client requests the

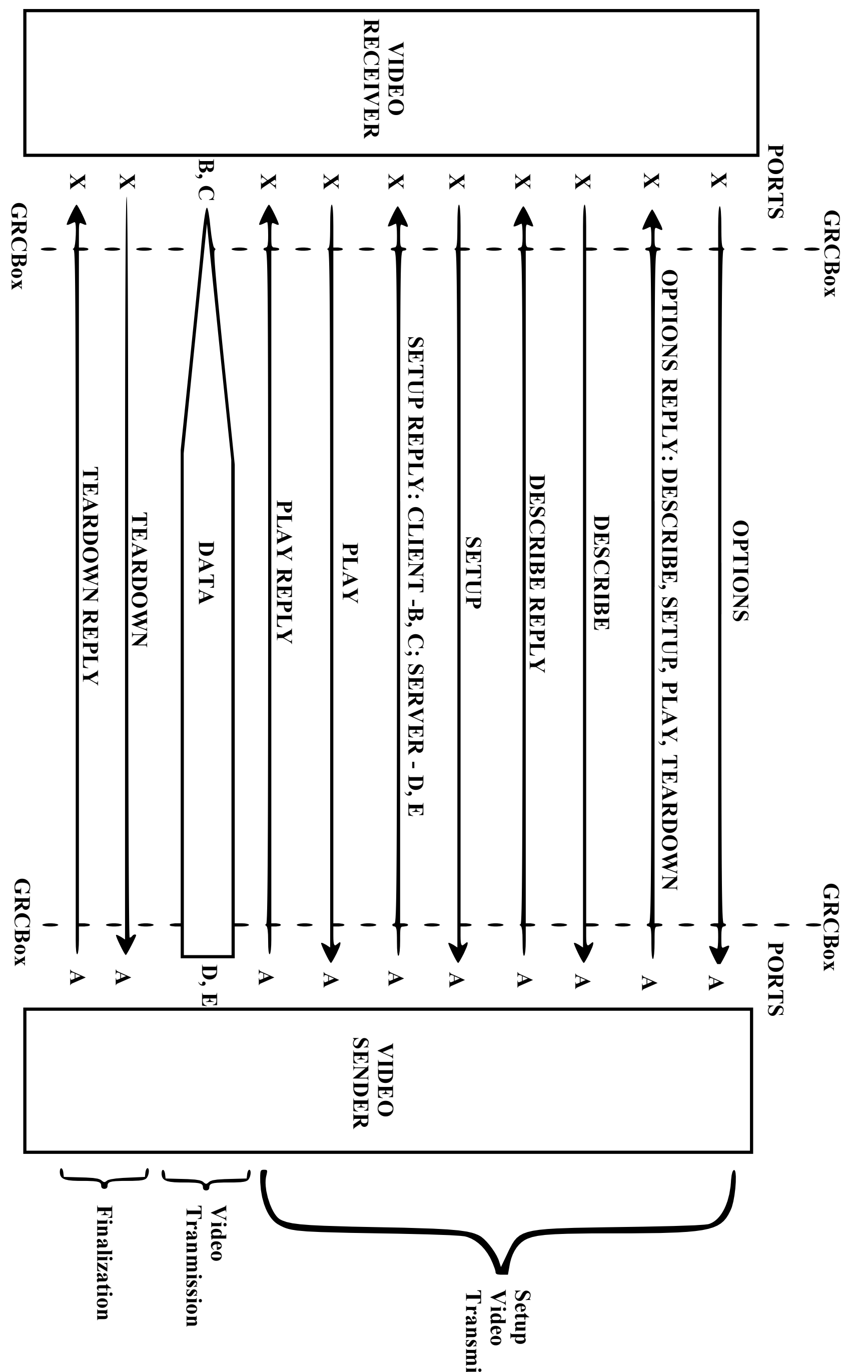


*Figure 4.11: The video transmission process: client-server message exchanges.*

server to start configuring the video streaming process, using the *setup* option. This results in the server replying with the port numbers to be used for the sending and receiving of the video, as well as audio data, if any. Note that our application only makes use of video data, but no audio. In this case, port numbers B and C are to be used for receiving, while D and E are used for sending. Hence, following the instructions provided by the server, the client tries to open ports B and C before sending the *play* request. The server acknowledges the *play* request by the client, and then, in Step-II, an exchange of video data takes place between the server and client using the previously negotiated ports. UDP is used for the sending of audio and video data, while in the other steps, messages are exchanged using TCP. To stop video streaming, the client initiates a *teardown* request, to which the server acknowledges, causing the video streaming to end.

### 4.2.4. Creating the vehicular network

For proper operation, the developed application assumes the availability of a vehicular network. Although, the vehicles we use on a daily basis still lack the capability to communicate with one another. So, for testing EYES, we equipped cars with GRCBoxes (discussed in the previous chapter) inside them. GRCBox enables the integration of smartphones to vehicular networks. It was developed mainly due to the difficulty in creating an adhoc network using smartphones. Another important feature provided by GRCBox is the support for V2X communications. The different networks supported by the GRCBox include adhoc, cellular and wifi access points, among others. Thus, we use the adhoc network support of the GRCBoxes to create the required network for our application.

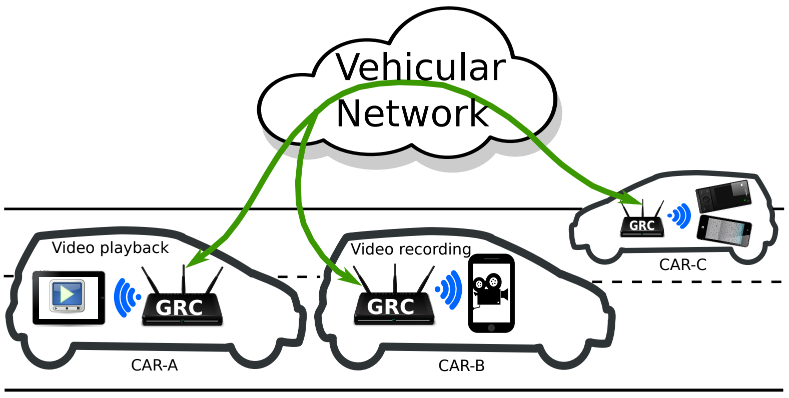


*Figure 4.12: Our application working together with GRCBox.*

Figure 4.12 shows how EYES works when combined with GRCBox. Each car within the vehicular network has a GRCBox mounted. The smartphones of the

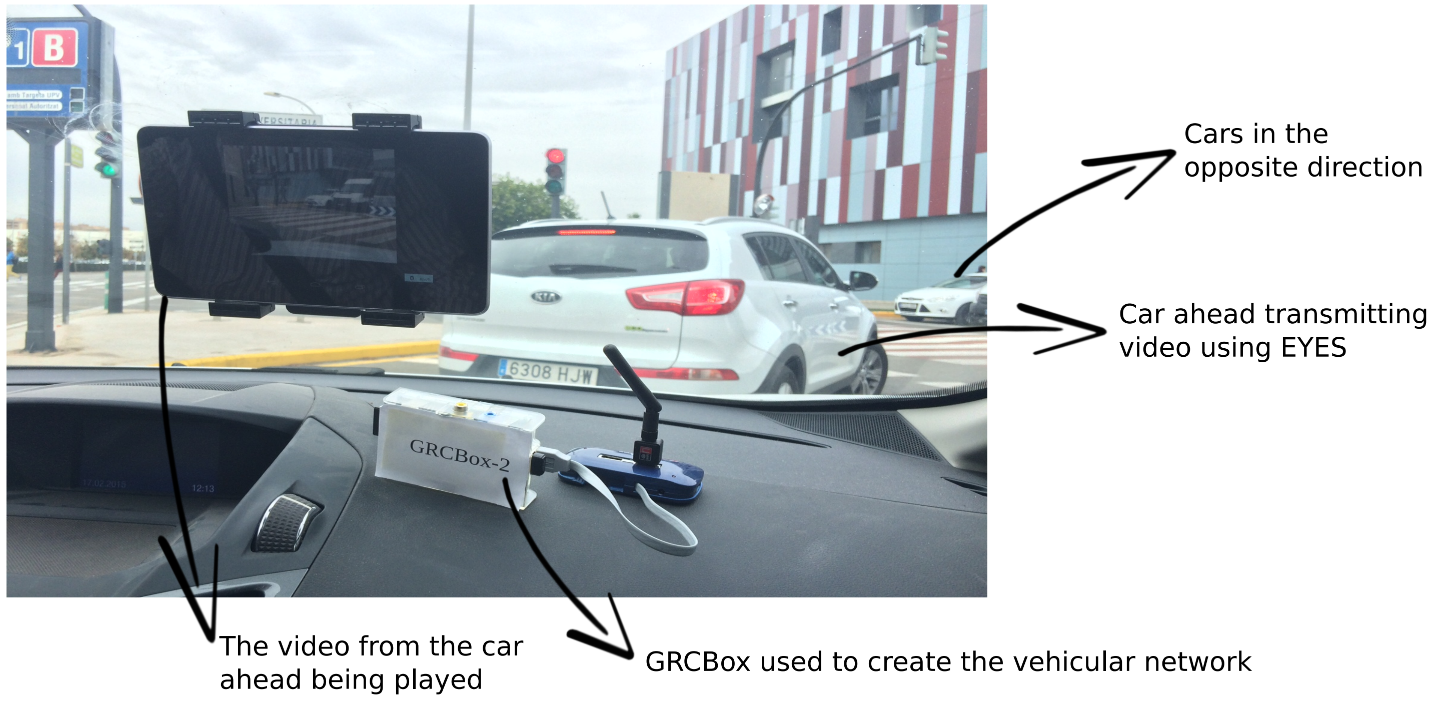

Cars in the opposite direction
Car ahead transmitting video using EYES
The video from the car ahead being played
GRCBox used to create the vehicular network
GRCBox-2
6308 HJW

*Figure 4.13: The experiments with EYES in real scenario.*

passengers within the car are connected to the GRCBox, which is equipped with wifi-enabled USB interfaces to communicate in adhoc mode, creating a vehicular network. Even though each GRCBox is supposed to be equipped with 802.11p devices for vehicular communication, we used 802.11a devices instead, as 802.11p-enabled hardware was not available while setting up the GRCBox to perform the tests. In future experiments we intend to use 802.11p compatible hardware to take advantage of the WAVE standard.

As shown in figure 4.12, Car-B is ahead of the Car-A, and both of them are travelling in the same direction and running our application, so the smartphone in Car-B starts recording the video autonomously and sending it to Car-A, relying on the vehicular network created using the GRCBoxes available within the cars. Concerning the video, it is played onscreen on the device in Car-A as soon as video reception starts.

Figure 4.13 shows a photograph taken during one of the outdoor tests with the EYES application. In this picture, we can see our application taking advantage of the GRCBox for delivering video aid to the driver of the car, depending on which he or she might decide the safe moment to start the overtaking manoeuvre. Here, the car in front is trying to take a right turn, and the car behind is receiving the video from the car ahead and playing it onscreen, enabling to see what the driver in the car ahead is observing at that instant.

## 4.3. Application Validation

In this section, we are going to present the results obtained using our application. The application was deployed for validation in both laboratory and outdoor scenarios at the Technical University of Valencia, Spain, and Ghent University, Belgium, respectively. The experiments done in the laboratory environment aimed at selecting the best video codec and performance settings for our application. Once the application parameters were selected from our laboratory experiments, we went ahead to test the application outdoors in real scenarios.

### 4.3.1. Laboratory Evaluation

During application development, Android supported the encoding of video in H.263 [194] and H.264 formats, and images in the JPEG and Portable Network Graphics (PNG) [195] formats. We decided to use H.264 and the simpler MJPEG codecs in our drive safety application, and compare their performance in terms of how the video quality is affected for both of them in the presence of packet losses, as well as the delay involved between capture and playback. Based on the obtained results, we choose the most relevant codec for our real-time video-based overtaking application.

#### 4.3.1.1. Video Quality Experiments

For the first set of experiments in the laboratory environment we determined how the quality of the video is affected when there are losses. The idea here is to select the video codec that is least affected by losses, because our application is to be used in a wireless environment with nodes in constant motion, and so the communication between the nodes will be affected by information losses.

The first metric that we are going to use to compare the two video codecs is PSNR [196], which is the ratio between the maximum possible power of a signal and the power of the distorting noise that affects the quality of representation of the original signal.

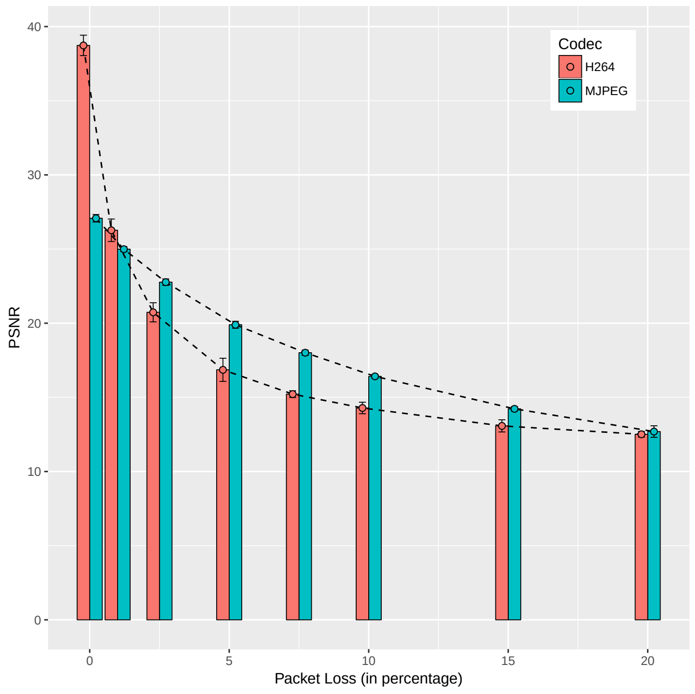


*Figure 4.14: Variation of PSNR with Packet loss for H.264 and MJPEG.*

Figure 4.14 shows how the PSNR for the video produced using H.264 and MJPEG is affected by the presence of packet losses. It can be seen from the graph that the percentage of packet losses was varied from 0 to 20 percent. A loss of 0 percent means the receiver obtains exactly the same video that the sender transmits, in which case the PSNR should be theoretically undefined. But since a comparison is made between the received video stream and the original RAW data, which is encoded before being sent, the PSNR value is never undefined. For the case when there are no losses, it can be noted that H.264 performs better than MJPEG. The reason for this phenomenon is that MJPEG is a much simpler codec compared to H.264, and so there is no inter-frame compression involved, resulting in an output video that occupies more space. The encoder for MJPEG tries to reduce the network usage by producing a video of lesser quality than the one produced by the H.264 encoder, resulting in better PSNR values in case of H.264. However, as packet loss appears in the scenario, we can see that H.264 is more affected than MJPEG, as PSNR values fall more steeply for H.264, which is due to its dependency on inter-frame compression.

Another important metric for the comparison of image or video quality is the SSIM [197, 198] index. SSIM is a method for measuring the perceived similarity between two images, and it is designed to improve upon traditional methods such as PSNR and Mean Squared Error (MSE) [199]. Similar experiments are repeated with H.264 and MJPEG once again, but this time to calculate the SSIM values for each of the video codecs.

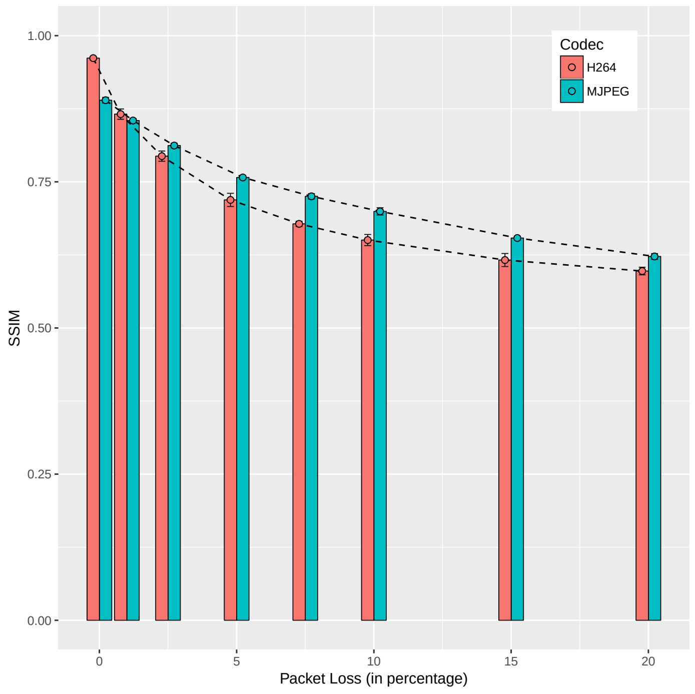


*Figure 4.15: Variation of SSIM with Packet loss for H.264 and MJPEG.*

Figure 4.15 compares the SSIM values for H.264 and MJPEG, and how it is affected by packet losses. This figure shows similar trends as in figure 4.14, but in this case the SSIM values decline more smoothly in the presence of data loss. It is important to note that, according to the SSIM index, the difference in the quality of the initial videos produced from the RAW data by the two video encoders is not that significant, being that the video produced by the H.264 encoder is only sightly better. However, as packet loss increases, the quality of the H.264 video is more affected than in the case of MJPEG.

Thus, the inference from our video quality experiments is that MJPEG is a better choice as the video codec to be used in the proposed application for our high mobility vehicular scenarios, especially when considering that it is more resilient to packet losses than H.264.

## 4.3.2. Application Delay Experiments

An important parameter to determine the functionality of our application, which is characterised by providing a visual aid to drivers to assist them in overtaking, is the delay between video capture and its playback. If this delay is too high, the video played at the receiver end would be of no real use to the driver. Thus, we have to first calculate the maximum admissible delay that we can afford, and then find out if the delay requirement is fulfilled by the different resolutions of our chosen MJPEG compression scheme, as well as the H.264 video codec.

For calculating the maximum allowable delay for our application, it is important to have an idea of the safe overtaking distance between the car trying to overtake and the car coming from the opposite direction. This is because the application delay would cause the car coming from the opposite direction to be actually closer to the overtaking vehicle than it appears.

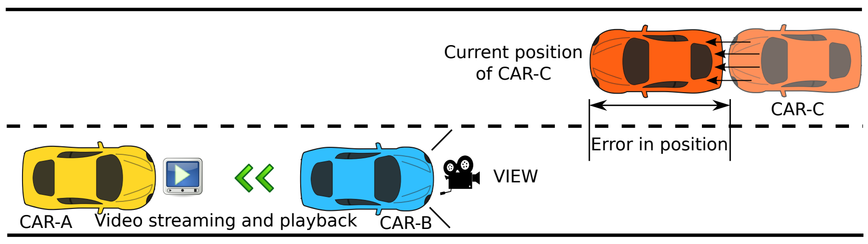


*Figure 4.16: Error due to delay.*

Figure 4.16 explains the idea that the vehicle coming from the opposite direction, in this case CAR-C, would be closer than it is seen in the displayed video at CAR-A, due to the delay in between capture at CAR-B and its playback at CAR-A. While studying the explained phenomenon, Crawford [200] found that, under ideal conditions, a distance of 228.6 meters is required for overtaking at about 80.47 km/h. However, Hills [201] further showed that, for both overtaking and incoming vehicle speeds of 80.47 km/h, the total overtaking distance required is of the order of 457.2 meters, twice the distance recommended by Crawford.

Our application has been designed to be used majorly while driving on single carriageways where the road is undivided and has traffic moving in both directions with high velocity. Now, let us assume that we have two cars moving in the opposite direction at 80.47 km/h (similar to the assumptions made by Hills). The *relative velocity* ($V_R$) can be calculated using the formula:

$$V_R = V_A + V_B$$

where $V_A$ and $V_B$ are the velocities of the two cars, and $V_R$ is found to be 160.94 km/h or 44.706 m/s. Limiting the maximum error in the positioning of the vehicle coming from the opposite direction to 5 percent of the total overtaking distance of 457.2 meters as suggested by Hills, would allow an error of 22.86 meters. Now, considering the maximum error in positioning as 22.86 meters, and a relative velocity of 44.706 m/s, the *maximum allowable delay* for our application is *0.511 seconds* in accordance with the equation: $time = distance/speed$.

In the next experiment with the two codecs, we will see whether both of them fulfil our strict delay requirement. It is to be noted that fulfilling the time constraint would make our application usable even while driving on roads with a higher speed limit than the assumed speed of 80.47 km/h. This is because vehicles driven at a higher speed would also proportionally need a larger overtaking distance from the vehicle coming in the opposite direction. Compared to H.264, MJPEG is a simpler video compression format since the video stream is compressed separately as JPEG images. Thus, when talking about compression-ratios, the performance of

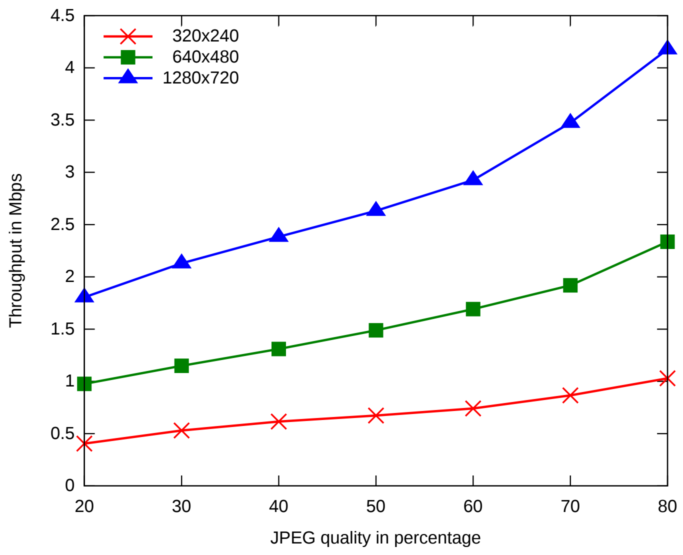


*Figure 4.17: Variation of throughput with JPEG quality for a 10 FPS MJPEG video.*

MJPEG is limited. So, to make a comparison between H.264 and MJPEG encoding schemes for Android devices in terms of delay, we first calculate the throughput of MJPEG video for different resolutions, so that we can eventually make delay versus throughput comparisons for the two encoding formats.

In Figure 4.17, the frames per second of the MJPEG video stream was fixed at 10 because we believe that a 10 FPS video is sufficient for our application. Also, the JPEG compression function used to generate the frames required for the video feed accepts a value between 0 and 100. The value 0 produces the worst perceived quality, but the highest compression, while the opposite occurs for 100. For our experiments, the quality of the JPEG in the video stream was varied from 20 to 80 percent, since for lower values the video quality was too low, whereas a JPEG quality of more than 80 percent did not show any significant improvements in the perceived quality. From the figure we can observe that, for a resolution of 320x240, the average throughput varies from 0.405 to 1.029 Mbps. For 640x480, it lies between 0.976 and 2.336 Mbps, and in case of 1280x720 resolution, it ranges between 1.805 and 4.177 Mbps.

We now supply the throughput values we achieved for MJPEG to the H.264 encoder to make a comparison between them, and to obtain the delay for the two types of encoding formats for different resolutions.

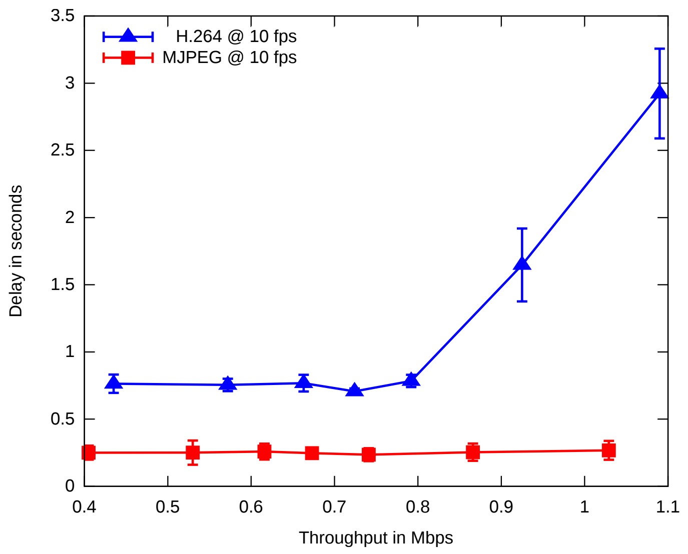


*Figure 4.18: Delay comparisons for 320x240 MJPEG and H.264 video streams.*

Figure 4.18 allows observing that, for a resolution of 320x240, the average delay for MJPEG suffers minimal variations (from 0.24 to 0.27 seconds), whereas for H.264, it increases from 0.71 to 2.92 seconds.

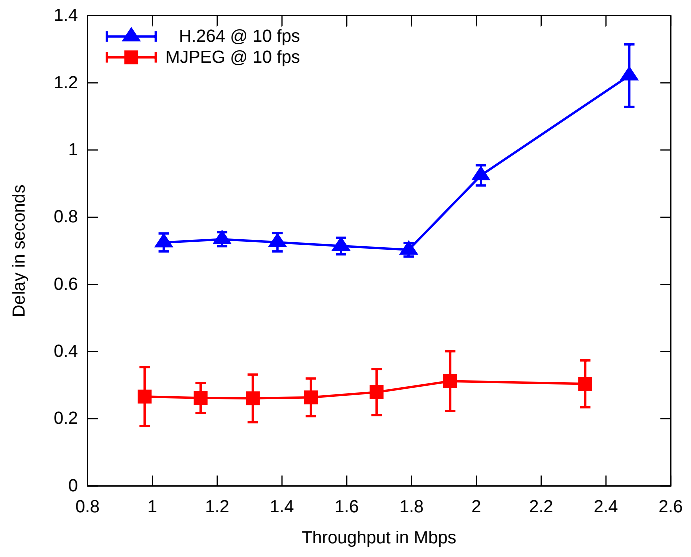


*Figure 4.19: Delay comparisons for 640x480 MJPEG and H.264 video streams.*

Similarly, figure 4.19 shows that the average delay ranges from 0.26 to 0.31 seconds for MJPEG, and from 0.7 to 1.22 seconds for H.264 video, the resolution being 640x480 for both the encoding formats.

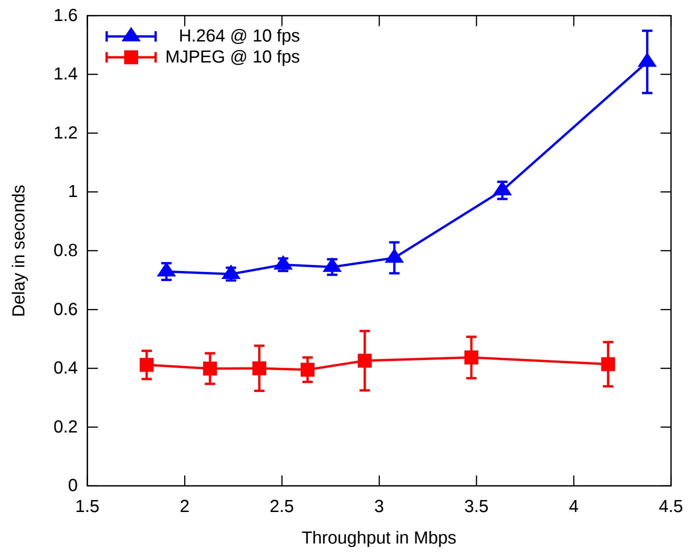


*Figure 4.20: Delay comparisons for 1280x720 MJPEG and H.264 video streams.*

Eventually, in figure 4.20, which compares H.264 with MJPEG for a resolution of 1280x720, we see that, in case MJPEG is used, the mean delay ranges between 0.4 and 0.44 seconds, and it is in the range from 0.72 to 1.44 seconds for H.264. Thus, in all the cases, MJPEG outperforms H.264 when considering delay in scenarios involving Android devices. Now, keeping in mind the delay requirement we have set previously, we are able to use any resolution of up to HD for MJPEG video encoding.

#### 4.3.2.1. Chosen Video Settings

Next, we want to select the most appropriate resolution and JPEG quality for the MJPEG video stream for use in the scope of our application. The proper functioning of the application is largely dependent on the availability of a vehicular network, which has been created using GRCBoxes. Thus, this selection process depends on the bandwidth provided by the vehicular network. From our experiments with the GRCBox, we found that it is capable of providing a mean bandwidth of 10.5 Mbps for TCP traffic, and 15.5 Mbps for UDP traffic, although the worst value for both TCP and UDP was close to 5.5 Mbps. Since an Android device with our application installed can simultaneously act as video source and destination, the effective bandwidth available for one-way video transmission in the worst-case scenario is 2.75 Mbps. At a data rate of 2.75 Mbps, all the different combinations of resolution up to HD with JPEG quality up to 50 percent, as suggested by Figure 4.17, can be supported by the vehicular network created using GRCBoxes; consequently, we chose the MJPEG compression scheme with a resolution of HD at 10 FPS with JPEG quality set to 50 percent for the video stream, owing to its better performance in terms of delay, while meeting throughput limitations.

### 4.3.3. Outdoor tests

Now, using the chosen setting from our indoor experiments carried out in the laboratory environment, we performed some simple tests in the parking area of the Ghent University Schoonmeersen Campus. The experiments were performed on very straight roads of two lanes exploiting the fact that overtaking will usually be performed on straight segments of the road, rather than at curves or bents. During our experiments, we evaluated the conditions that are a part of the *validity check* on which our EYES application depends.

#### 4.3.3.1. Same Direction Test

As we have already seen from section 4.2.2, the *Same Direction Test* is used to check if the two vehicles are travelling in the same direction, and is dependent on a threshold value. The value of this threshold was established at 12.5 degrees, from experiments performed with our Android FCW application presented in the previous chapter.

Next, we want to verify when one vehicle follows the other, travelling in the same direction and on the same lane, if the established threshold of 12.5 degrees would always hold in the context of this test. Note that, since regression is performed to correct lateral errors of the GPS positions before constructing the displacement vectors of the vehicles, we have made a comparison of Least Absolute Deviations (LAD) [202], Linear, Least Trimmed Squares (LTS) [203], Repeated Median [204], and Single Median or the Theil–Sen estimator [205] regression models.

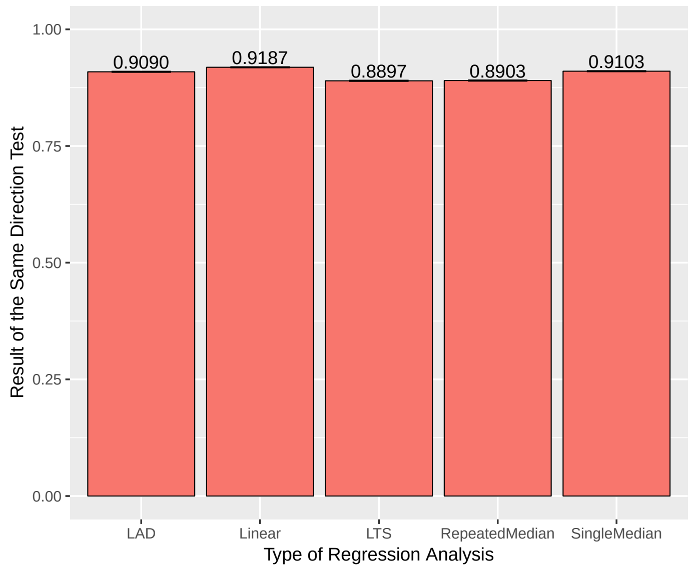


*Figure 4.21: Results of the Same Direction Test when both the vehicles were moving on the same lane and direction.*

Figure 4.21 shows the result from the *Same Direction Test* when the vehicles were actually moving in the same direction. A value equal to 1 signifies a positive result from the test all the time, while 0 means a negative outcome from the test. Here the ground truth was known since the vehicles were indeed moving in the same direction, thus we can see that Linear regression gives the best result of 0.92, followed by Single Median and LAD with 0.91, while in the case of Repeated Median and LTS, the test achieves success 89 % of the times. Thus, we observe similar performance from all the five different types of regression methods considered here, with failure rate between 8 to 11% of the times, even when the vehicles

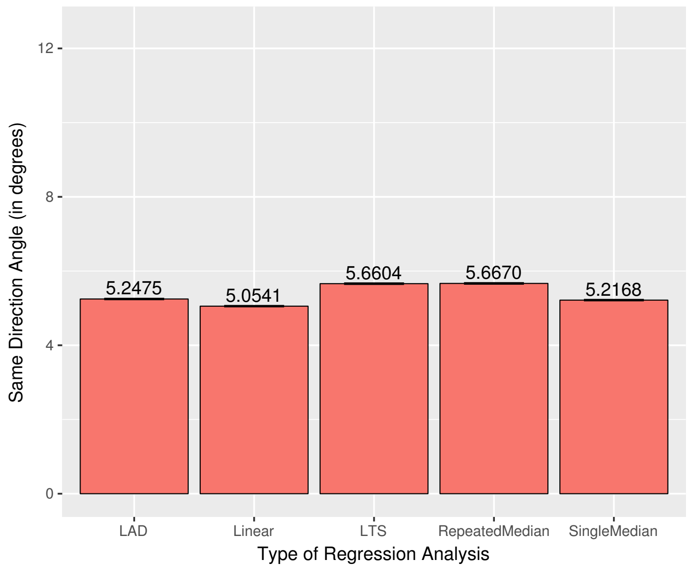


*Figure 4.22: Angles measured by the Same Direction Test when both the vehicles were moving on the same lane and direction.*

are known to be travelling in the same direction.

Figure 4.22 depicts the angles measured by the *Same Direction Test* when the vehicles were moving in the same direction. We already set the threshold of this test to 12.5 degrees in accordance with previous experiments. The aim was to see if the values achieved were within this predefined threshold. As can be seen from the figure, angles measured using the different regression methods were well below the threshold value.

Next, we wanted to study the effect of the length of displacement vectors and inter-vehicular distance on this test. For this reason, we define a ratio $(d_t/d_c)$ of the length of the displacement vector of the vehicle behind $(d_t)$, and the distance from the vehicle behind to the projection of the current corrected location of the vehicle ahead on the displacement vector of the vehicle behind $(d_c)$. Figure 4.23

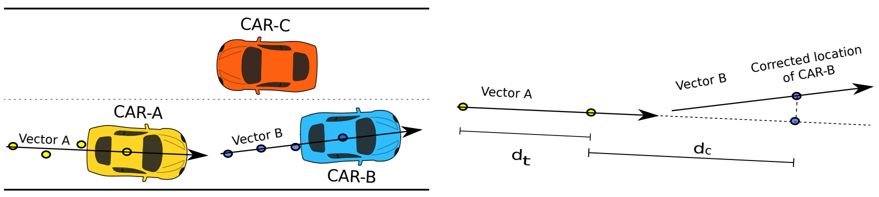


*Figure 4.23: Definition of $d_t$ and $d_c$.*

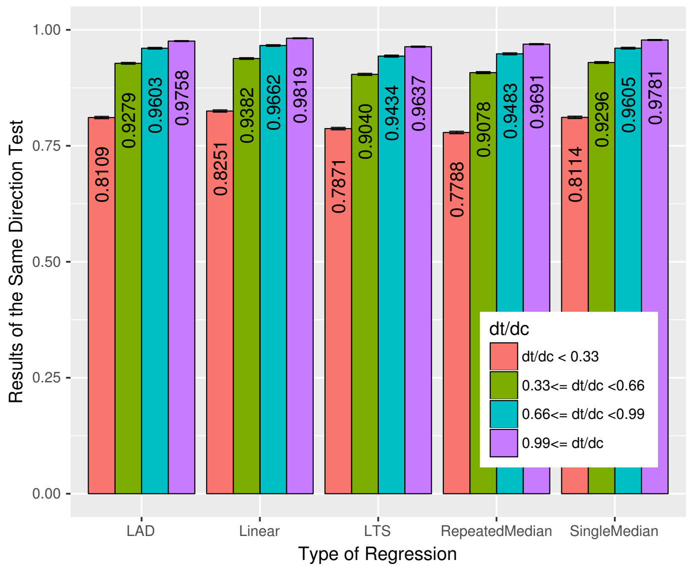


*Figure 4.24: Results of the Same Direction Test grouped according to the ratio of $d_t/d_c$ when the vehicles are travelling in the same direction.*

explains the defined terms.

Figure 4.24 shows the performance of the *Same Direction Test* grouped according to the ratio of $d_t/d_c$. We observe that the Linear and the Single Median regressions perform better than the rest, and with higher values of $d_t/d_c$ the test produces better results. This means that if the displacement vector of the vehicle behind is sufficiently large, and the inter-vehicular distance is small, the *Same Direction Test* provides better results. In case of Linear regression the efficiency of this test rises from 83 to 98%, while it varies from 0.81 to 0.98 for Single Median. This means that, as the car behind slowly approaches the vehicle in front, this test used by our application will show improved performance.

Figure 4.25 illustrates the angles measured by the *Same Direction Test* when the two vehicles were travelling in the same direction and on the same lane. It can be seen that the angles measured are well below the threshold limit of 12.5 degrees, and with higher values of $d_t/d_c$, the angles measured are also much lower.

### 4.3.3.2. Ahead Test

The *Ahead Test* is used to detect if a possible source of video is actually ahead of the video receiver vehicle. In the experiments with this condition, we used two vehicles: one ahead and the other following it on the same lane, at all times. The aim of the experiment was to examine the performance of this test given a known

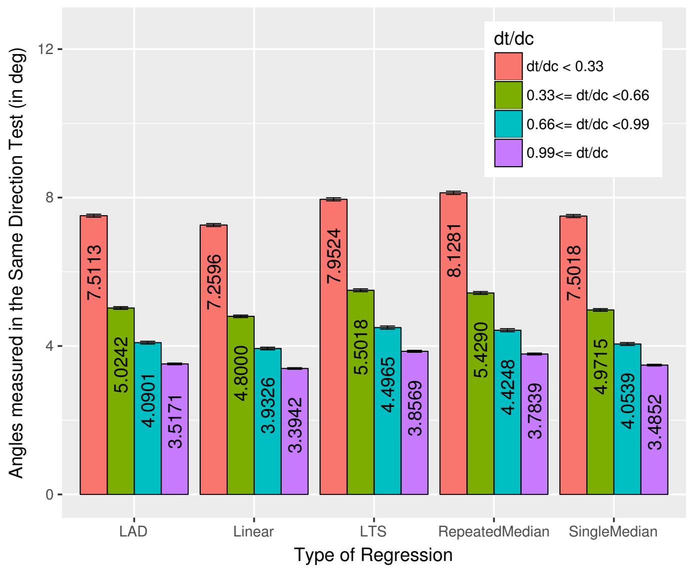


*Figure 4.25: Angles measured by the Same Direction Test grouped according to the ratio of $d_t/d_c$ when the vehicles are travelling in the same direction.*

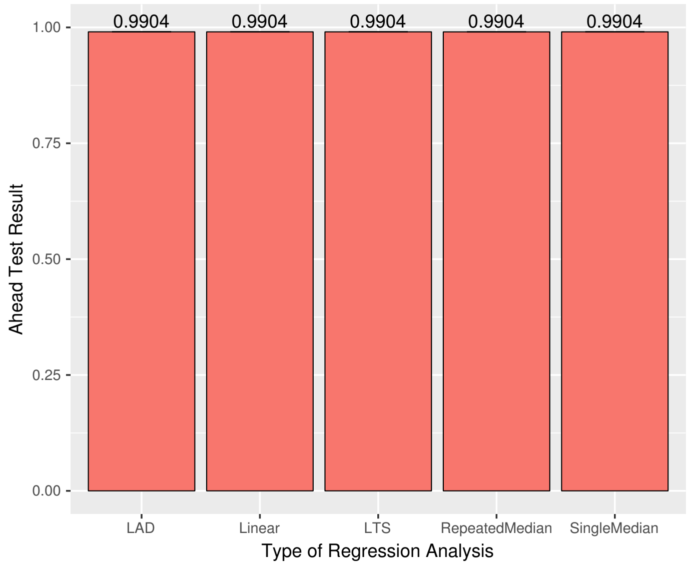


*Figure 4.26: Result from experiments with the Ahead Test when one vehicle is ahead of the other.*

ground truth. The value of 1 represents that all outcomes of this test were positive, while on the contrary, a value of 0 implies that the test failed in all of the cases considered.

From figure 4.26 we can see that the different types of regressions, namely LAD, Linear, LTS, Repeated Median, and Single Median show a very similar performance. We also find that, over 99% of the times, it is possible to efficiently identify if the vehicle that wishes to stream video is actually travelling ahead of the video destination.

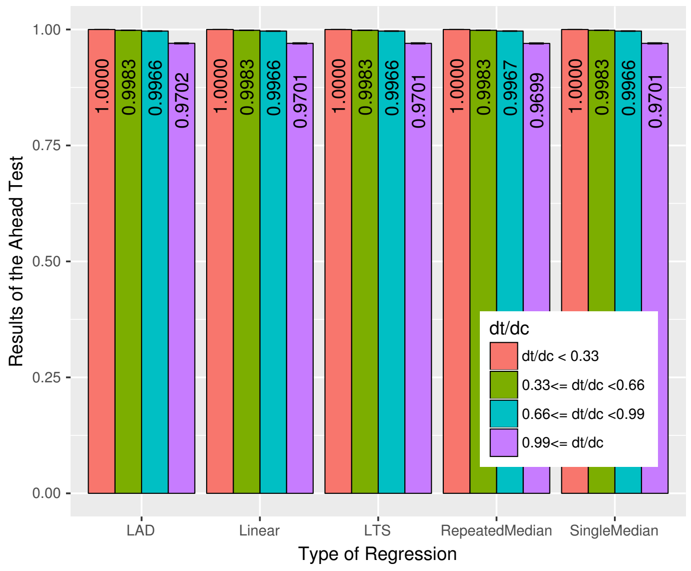


*Figure 4.27: Result from experiments with the Ahead Test grouped according to the ratio of $d_t/d_c$ when one vehicle is ahead of the other.*

Figure 4.27 confirms that the accuracy in detection of a vehicle travelling ahead of the other is reduced with the increase in the value of $d_t/d_c$. As can be noted from the figure, the accuracy of the test drops from 100 to 97% as the value of ratio $d_t/d_c$ increases from less than 0.33, to 0.99 and above. This is because, when both the vehicles are far away from each other, it is easier to detect if the other vehicle is ahead, but as the vehicle behind moves closer to the vehicle ahead with the intention of overtaking, the detection becomes more difficult, and so the *Ahead Test* fails more often in that case.

### 4.3.3.3. Same Lane Test

This test is used to ascertain if the vehicles between which the video streaming would take place, are actually travelling on the same lane.

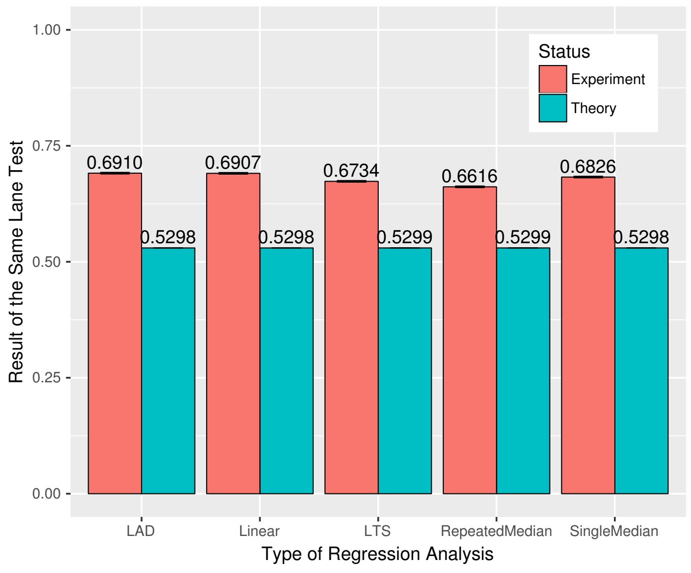


*Figure 4.28: Result from experiments with the Same Lane Test when both vehicles were on the same lane.*

Figure 4.28 shows the results obtained in our experiments with the *Same Lane Test*. It can be seen that LAD, and Linear regressions, obtain the best outcome with the average of 0.69. They are followed by Single Median, LTS and Repeated Median with averages of 0.68, 0.67, and 0.66 respectively. Note that, in this case, the two vehicles were actually on the same lane, and thus higher values for this experiments means better performance, with the maximum possible value of 1 indicating that all the cases were properly detected. Here, a maximum of 69% of the cases were detected by the *Same Lane Test*. Also, it performs better than what the theory of the test (equation 4.1) suggests, because in the theory two GPS points are considered with similar uncertainty or sigma values, but in our experiments, we have performed regression considering multiple points. In our preliminary analysis of the *Same Lane Test* we had considered the construction of displacement vectors with just two GPS locations, and in that case, the experimental results were in accordance with what the theory suggests. However, since the uncertainty due to the large inaccuracy in GPS data resulted in detection of only about 50% of the cases when the vehicles were actually on the same lane, this led us to try and use regression in our efforts to improve the achieved results.

Figure 4.29 depicts the effect of $d_t/d_c$, which is the ratio of the length of the displacement vector of the vehicle behind ($d_t$), and the distance from the vehicle behind to the projection of current corrected location of the vehicle ahead on the

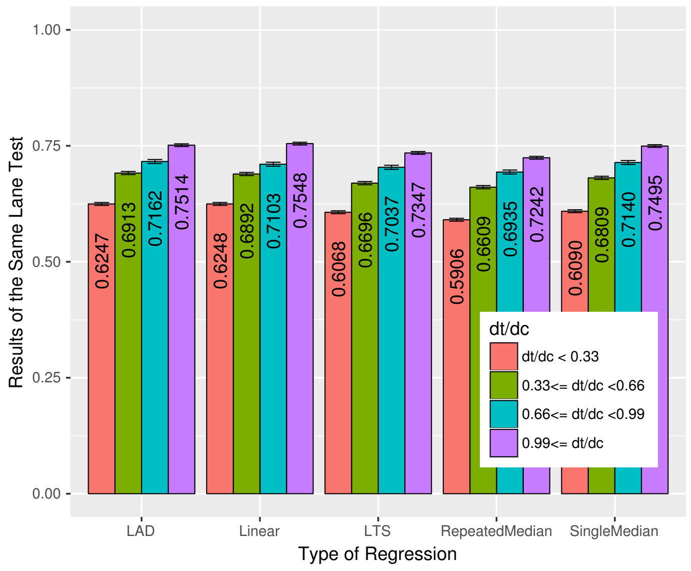


*Figure 4.29: Result from experiments with the Same Lane Test grouped according to the ratio of $d_t/d_c$ when both vehicles were on the same lane.*

displacement vector of the vehicle behind ($d_c$). We can observe from the figure that, with an increase in the value of $d_t/d_c$ from below 0.33 to more than 0.99, the detection rate of our *Same Lane Test* increases from 0.63 to 0.75 for the LAD and Linear regression methods. In addition, it rises from 0.61 to 0.75 for the case of Single Median, ascending from 0.61 to 0.73 for LTS, and for Repeated Median, it improves from 0.59 to 0.72.

In the next part of the study of our *Same Lane Test*, we will check its accuracy when the two vehicles are actually travelling on different lanes. This way we can confirm the usability of this test as it has been designed to actually detect cases when overtaking is occurring, and the vehicle which was previously behind, is now located on the other lane, ahead of the vehicle being followed. In such situations, this test should make sure that video streaming does not start from the overtaking vehicle to the vehicle that was previously ahead, unless the overtaking vehicle completes the action and reaches its new position on the same lane.

Figure 4.30 allows observing that, when vehicles were travelling on different lanes, the designed test that evaluates if they are on the same lane generates positive results about 34% of the times when Single Median, Linear, LTS, and Repeated Median regression analysis is employed. Whereas for LAD, the *Same Lane Test* detects false positives in 35% of the cases. In this test, lower values are considered better as vehicles were travelling on separate lanes. The results achieved here are

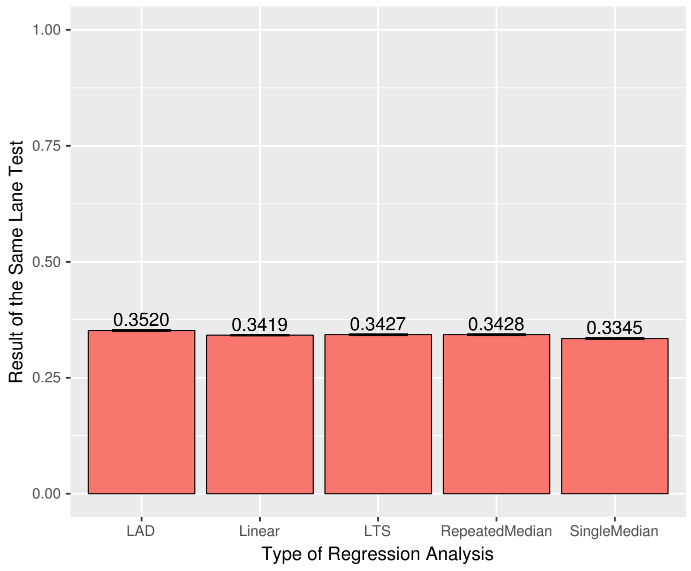


*Figure 4.30: Result from experiments with the Same Lane Test when vehicles were travelling on different lanes.*

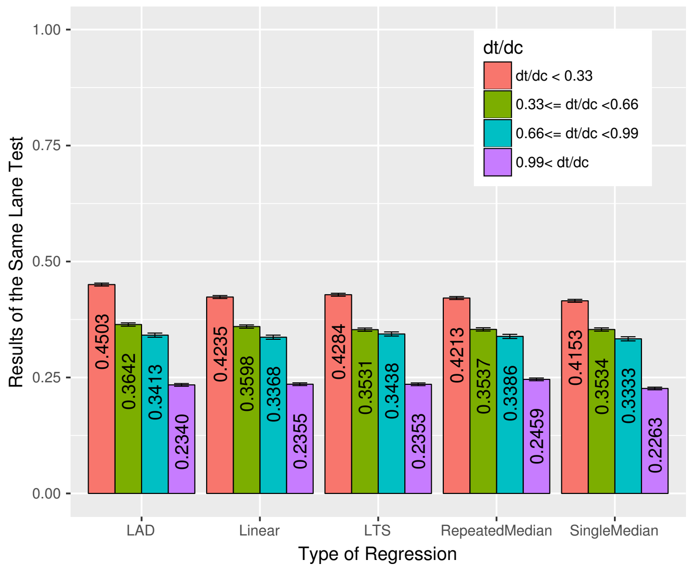


*Figure 4.31: Result from experiments with the Same Lane Test grouped according to the ratio of $d_t/d_c$ when vehicles were on separate lanes.*

well in accordance with the ones accomplished and illustrated in figure 4.28.

From figure 4.31 we can perceive that, with the variation of $d_t/d_c$ from below 0.33 to over 0.99, the detection of false positives drops from 0.42 to 0.23 for Single Median, which performs best in this case. Regarding other regression methods, like LAD, Linear and LTS, erroneous detection by the *Same Lane Test* reduces from 0.45 to 0.23, 0.42 to 0.24, and 0.43 to 0.24, respectively. The worst performance is shown by Repeated Median, for which the detection rate changes from 0.42 to 0.25 for varying values of the $d_t/d_c$ ratio.

## 4.4. Conclusions

We have introduced an application called EYES, that is able to provide drivers with a real-time visual overtaking aid. The developed system makes use of smartphones to capture and display the video stream. The video provided by the vehicle ahead is visible without user intervention at the vehicle just behind it by making use of the smartphone display. Hence, our application provides drivers with useful information about the traffic situation ahead, based on which the decision to overtake can be taken. This is specially useful when the view of the driver is blocked by a larger vehicle ahead. Also, since the transmission of data takes place between the vehicle just ahead to the one following it, there is no multi-hop relaying required, which makes us optimistic regarding its scalability and use in regions with high traffic density. The developed application was tested in both laboratory and outdoor scenarios.

The tests performed within the laboratory involved choosing the best video settings for our application. In particular, it involved a comparison between H.264 and MJPEG video encoding formats; MJPEG was chosen as the default video compression scheme due to its simplicity, better performance under losses, and lower encoding delay. We have also introduced a *validity check* consisting of three conditions, namely the *Same Direction Test*, the *Ahead Test*, and the *Same Lane Test* to choose the most adequate video server and client, as well as to initiate or terminate video streaming. These conditions have been kept very simple to achieve low delays, and thus enhance the usefulness of our system. Out of these three tests, the *Same Direction Test* is dependent on a threshold which we had previously defined as 12.5 degrees, further experiments confirm that the threshold value is good enough, and that all observations were under this value. Regarding the *Ahead Test*, it showed gratifying performance, being able to detect 99% of the cases on average when a vehicle is ahead of the other. Furthermore, when the two vehicles are far apart from one another, to when the vehicle behind closes in on the vehicle ahead, its efficiency decreases from 100% detection rate to 97%. Nevertheless, the *Same Lane Test* was found to be more affected by the inaccuracy of GPS data, and was able to efficiently detect 69% of the cases when the vehicles were using the same lane. Nevertheless, on analysing the accuracy of this test based on the length of the displacement vectors and the inter-vehicular distances, we found that the performance of the test improved from 0.63 to 0.75 in the cases where LAD and Linear regression methods were employed.

# 5
# Overall Conclusion

## 5.1. Summary of contributions

The goal of our research was to develop and evaluate smartphone-based ITS applications for VNs, and in turn study the of impact integrating smartphones to VNs. Unfortunately, we found that the vehicles we use on a daily basis still lack the capability to communicate with each other, due the slow adoption by manufacturers, hence making this analysis difficult. In addition, even though smartphones have the hardware capability to communicate in adhoc mode, it is limited by its software since it was not designed for situations where adhoc communication would be necessary. To address this issue, we designed a credit card sized low cost connectivity box named GRCBox that supports V2X communication. Each vehicle that wishes to participate and take advantage of applications designed for VNs has be equipped with our GRCBox, which acts as a router for data transmission. This communication box allows users within the vehicle to connect to it, and choose which type of communication is to be used per session. Possible network options are wifi, cellular and adhoc networks. Users are allowed to add new network interfaces as per their requirement, thus supporting V2X communication.

We also designed three safety applications for Android-based devices that make use of the GRCBox for inter-vehicular communication and data exchange. The first application was named Messiah, that aims to help reduce the response time of emergency vehicles like ambulances, police cars and fire brigades. The Messiah application is a navigation application that also displays emergency notifications using inter-vehicular communication. Thus, with the help of these on-screen notifications, the driver of the vehicle can take decisions like leaving space for the emergency vehicle to pass, or rush to the aid of the other vehicle in need of assistance. Experiments using this application confirmed that it performs best in networks with a high node density.

We have also presented a FCW application that uses license plate recognition and vehicular communication to generate warnings for notifying the drivers of the vehicle behind and the one ahead, of a probable collision when the vehicle behind does not maintain a distance of a vehicle length (approximately 5 meters) between itself and the vehicle ahead. The application was tested in both static and mobile scenarios, and we found out that, despite mobile devices with faster processors tend to function better, even high end models took excessive time to recognise license plates. Moreover, the on-board optical sensor of the smart devices was not powerful enough to stabilise the images captured when in motion, and in conditions involving low light. Thus, impeded by limitations in hardware, we felt that we might still have to wait for more powerful devices of the future to take full advantage of this application.

Finally, we have contributed towards the development of EYES, a drive safety application that aims to provide real-time visual overtaking aid to drivers. The designed system makes use of the smartphone camera to capture video, send it to the device located in the vehicle behind using the GRCBox network, where it is displayed on-screen for the driver to consult before overtaking. All this takes place without any sort of user intervention once the application has been properly launched. This is specially useful when the view of the driver is blocked by a larger vehicle ahead. Also, no multi-hop relaying of data takes place as the video streaming occurs between the vehicle just ahead to the one following it, which makes us optimistic regarding its scalability and use in regions with high traffic density. In our laboratory test with the EYES application we compared the use of two video codecs, namely the H.264 and MJPEG encoding formats, for selecting the most relevant video codec for our application. MJPEG was chosen as the default encoding method due to its simplicity, better performance under losses, and lower encoding delay. The designed application is dependent on three validation tests (*Same Direction Test*, *Ahead Test*, and *Same Lane Test*) for selecting the video source and destination. Experiments performed with the application in real scenarios show that the *Same Direction Test* and the *Ahead Test* performed as desired. However, the *Same Lane Test* was affected by the inaccuracy of GPS readings, and was able to efficiently detect 69% of the cases when the vehicles were using the same lane. On analysing the accuracy of this *Same Lane Test* based on the length of the displacement vectors and the inter-vehicular distances, we found that the performance of the test improved from 0.63 to 0.75 in the cases where LAD and Linear regression methods were employed.

From our implementation and experimental study of smartphone-based safety applications for VNs, we could see that, although integration of smartphones to VNs opens a new horizon for ITS applications, they still have to face limitations in terms of both quality and accuracy of the on-board sensors available to the mobile devices. These constraints soon come to the surface on designing and using a more demanding application that requires heavy processing or high sensor accuracy. Thus, the use of smartphones for ITS applications may result in more affordable and easily adoptable solutions in some cases, but may not be a good choice for more demanding and critical safety-related applications.

## 5.2. Future work

With respect to the different open research issues that can be pursued as future lines of work under the framework of this thesis, are the following:

- GRCBox - The communication device which supports V2X communication has not yet been tested with 802.11p, since compatible interfaces were not available to us while experimenting with it. Thus, it is highly desirable to add support for WAVE-compatible hardware and analyse its performance. Another possible improvement is to add support for communication with On-Board Diagnostics (OBD) or Controller Area Network (CAN) buses to retrieve vehicle-related information, and make it available to applications. Also, the support for generic networking rules, like forwarding broadcast packets to all outer networks that GRCBox is connected to, or to choose the most stable or high speed network for data transmission, could be helpful for the applications.

- Messiah Application - This safety application, along with the other applications presented in this thesis, has only been tested in real scenarios using two vehicles, each equipped with a smart device with the installed application. Although we are optimistic about the scalability of our application, further experiments involving more nodes participating in the network would help us confirm our assumptions. Also, the application has only been developed for Android devices, porting it to other platforms like iOS would increase its adoption and use.

- FCW Application - This application uses plate detection to estimate the distance between two cars, and generates warnings if a defined minimum safe distance is not maintained. The application is designed to function in bidirectional two-lane roads, where there is movement of traffic in both directions, and only one lane per direction of movement is available. Since it is capable of detecting multiple plates from images, it may detect plates of cars coming from the opposite direction, or static parked cars, in addition to the plate of the car ahead. Plates not belonging to the car ahead are discarded without warning generation with the help of a test that we have designed, named the *Same Direction Test*. Although this test performs well, it does not take into account multiple lane roads, where cars would be travelling in the same direction on more than one lane. Therefore, situations might arise where unwanted warnings are generated on identification of the plate of a vehicle travelling on a different lane, but in the same direction, which is within the communication range of the vehicle using our application. This problem has been left as future work, and needs to be addressed.

- EYES Application - It is a real-time visual overtaking assistant, which is dependent on three validation tests, namely the *Same Direction Test*, the *Ahead Test* and the *Same Lane Test*, for selection of the video source and destination. As discussed previously, while the *Same Direction Test* and the *Ahead Test* performed as per our expectations, the *Same Lane Test* on the other hand, was more affected by GPS errors. It could efficiently detect 69% of the cases when the vehicles were using the same lane. However, the performance of the test improved from 0.63 to 0.75 in the cases where LAD and Linear regression methods were used to analyse the accuracy of this test based on the length of the displacement vector and the inter-vehicular distances. Methods to further improve the reliability of this test should be sought.

Apart from the discussed issues, the presented solutions may be affected by different types of application layer attacks. It is to be noted here that we have not used any extra security measures to protect our applications from the different security-related threats, which have also been left as future work as a likely improvement in the next versions. Thus, some of the possible attacks that might influence the performance of the designed application are denial of service [206], non-control-data corruption [207], malwares, and hacks. Denial of service can occur if a server is unable to respond to a legitimate client when it is occupied in communication with the attacker. Approaches against this type of attack are usually based on the use of public key cryptographic protocols, an example of which is [208]. Non-control-data corruption attacks, on the other hand, refers to the access and modification of user, user identity, configuration or decision-making data. Our applications have been designed for the Android operating system, based on the Linux environment, which provides developers the possibility to store data that is private to the application, thereby rendering non-control-data corruption attacks difficult to execute [209]. An example of malware could be a software that infects the original code by modifying the actual behaviour of the application. A solution to this security threat includes the use of Control-Flow Integrity [210], that implies embedding within software executables both the control-flow policy which is enforced during runtime, and the mechanism for that enforcement. Hacking is another security threat whereby the attacker would be able to modify the performance of the application and supply receivers with corrupted data. These types of problems can be easily dealt with by making the whole system embedded within the car, and by forcing the application to be proprietary software with manufacturer certification.

## 5.3. Publications

In this section we present a list of all the publications obtained as a result of our study in relation to this dissertation, consisting of 4 journal articles, 7 conference papers, and 1 demo presentation.

### 5.3.1. Journals

- Patra, Subhadeep, Peter Veelaert, Carlos T. Calafate, Juan-Carlos Cano, Willian Zamora, Pietro Manzoni, and Fabio González. "A Forward Collision Warning System for Smartphones Using Image Processing and V2V Communication." Sensors 18, no. 8 (2018).

- Hadiwardoyo, Seilendria A., Subhadeep Patra, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni. "An Intelligent Transportation System Application for Smartphones Based on Vehicle Position Advertising and Route Sharing in Vehicular Ad-Hoc Networks." Journal of Computer Science and Technology 33, no. 2 (2018): 249-262.

- Patra, Subhadeep, Carlos T. Calafate, Juan-Carlos Cano, Peter Veelaert, and Wilfried Philips. "Integration of vehicular network and smartphones to provide real-time visual assistance during overtaking." International Journal of Distributed Sensor Networks 13, no. 12 (2017).

- Tornell, Sergio M., Subhadeep Patra, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni. "GRCBox: extending smartphone connectivity in vehicular networks." International Journal of Distributed Sensor Networks 11, no. 3 (2015).

### 5.3.2. Conferences

- Hadiwardoyo, Seilendria A., Subhadeep Patra, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni. "An Android ITS Driving Safety Application Based on Vehicle-to-Vehicle (V2V) Communications." In Computer

Communication and Networks (ICCCN), 2017 26th International Conference on, pp. 1-6. IEEE, 2017.

- Patra, Subhadeep, Carlos T. Calafate, Juan-Carlos Cano, Peter Veelaert, and Wilfried Philips. “Determining the relative position of vehicles considering bidirectional traffic scenarios in VANETS.” In Proceedings of the 2nd Workshop on Experiences in the Design and Implementation of Smart Objects, pp. 6-10. ACM, 2016.
- Tornell, Sergio M., Subhadeep Patra, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni. “A novel on-board unit to accelerate the penetration of ITS services.” In Consumer Communications & Networking Conference (CCNC), 2016 13th IEEE Annual, pp. 467-472. IEEE, 2016.
- Patra, Subhadeep, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni. “An ITS solution providing real-time visual overtaking assistance using smartphones.” In Local Computer Networks (LCN), 2015 IEEE 40th Conference on, pp. 270-278. IEEE, 2015.
- Patra, Subhadeep, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni. “Performance Tuning Of A Smartphone-Based Overtaking Assistant.” In XXVI Jornadas de Paralelismo, pp. 139-145. 2015.
- Patra, Subhadeep, Javier H. Arnanz, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni. “EYES: A novel overtaking assistance system for vehicular networks.” In International Conference on Ad-Hoc Networks and Wireless, pp. 375-389. Springer, Cham, 2015.
- Patra, Subhadeep, Sergio M. Tornell, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni. “Messiah: an ITS drive safety application.” In XXV Jornadas de Paralelismo, pp. 409-413. 2014.

### 5.3.3. Others: Demo

- Patra, Subhadeep, Sergio M. Tornell, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni. “Video-based overtaking assistance now a reality.” In Proc. 40th IEEE Conf. Local Comput. Netw.(LCN). 2015.

# References

[1] World Health Organization. *Global status report on road safety 2015*. World Health Organization, 2015.

[2] Car 2 Car Communication Consortium official website. `https://www.car-2-car.org/`. Accessed: 2019-06-01.

[3] Elyes Ben Hamida, Hassan Noura, and Wassim Znaidi. Security of cooperative intelligent transport systems: Standards, threats analysis and cryptographic countermeasures. *Electronics*, 4(3):380–423, 2015.

[4] Mohammad Almalag. Safety-related vehicular applications, 2009.

[5] Uichin Lee, Ryan Cheung, and Mario Gerla. Emerging vehicular applications. *Vehicular Networks: From Theory to Practice. Chapman and Hall/CRC*, 2009.

[6] EDUCBA tutorials. `https://www.educba.com/structure-of-an-android-operating-system/`. Accessed: 2019-06-02.

[7] Daniel Sperling and Deborah Gordon. *Two billion cars: driving toward sustainability*. Oxford University Press, 2010.

[8] A Aroussi, A Hassan, B Clayton, BS AbdulNour, and E Rice. Improving vehicle windshield defrosting and demisting. Technical report, SAE Technical Paper, 2000.

[9] Anders Lie, Claes Tingvall, Maria Krafft, and Anders Kullgren. The effectiveness of electronic stability control (ESC) in reducing real life crashes and injuries. *Traffic injury prevention*, 7(1):38–43, 2006.

[10] Kristy B Arbogast, Dennis R Durbin, Rebecca A Cornejo, Michael J Kallan, and Flaura K Winston. An evaluation of the effectiveness of forward facing child restraint systems. *Accident Analysis & Prevention*, 36(4):585–589, 2004.

[11] George Dimitrakopoulos and Panagiotis Demestichas. Intelligent transportation systems. *IEEE Vehicular Technology Magazine*, 5(1):77–84, 2010.

[12] Hassnaa Moustafa and Yan Zhang. *Vehicular networks: techniques, standards, and applications*. Auerbach publications, 2009.

[13] Yu Wang and Fan Li. Vehicular ad hoc networks. In *Guide to wireless ad hoc networks*, pages 503–525. Springer, 2009.

[14] Subir Biswas, Raymond Tatchikou, and Francois Dion. Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety. *IEEE communications magazine*, 44(1):74–82, 2006.

[15] Javier Gozálvez, Miguel Sepulcre, and Ramon Bauza. IEEE 802.11 p vehicle to infrastructure communications in urban environments. *IEEE Communications Magazine*, 50(5), 2012.

[16] Adam Thierer and Ryan Hagemann. Removing roadblocks to intelligent vehicles and driverless cars. *Wake Forest JL & Pol'y*, 5:339, 2015.

[17] Daniel Jiang and Luca Delgrossi. IEEE 802.11 p: Towards an international standard for wireless access in vehicular environments. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 2036–2040. IEEE, 2008.

[18] ASTM E2213-03(2010). Standard Specification for Telecommunications and Information Exchange Between Roadside and Vehicle Systems — 5 GHz Band Dedicated Short Range Communications (DSRC) Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *ASTM International*, 2010.

[19] Roberto A Uzcátegui, Antonio Jose De Sucre, and Guillermo Acosta-Marum. Wave: A tutorial. *IEEE Communications magazine*, 47(5), 2009.

[20] Vinod Kone, Haitao Zheng, Antony Rowstron, and Ben Y Zhao. On infostation density of vehicular networks. In *Wireless Internet Conference (WICON), 2010 The 5th Annual ICST*, pages 1–9. IEEE, 2010.

[21] IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments. *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009)*, pages 1–51, July 2010.

[22] IEEE Draft Guide for Wireless Access in Vehicular Environments (WAVE) - Architecture. *IEEE P1609.0/D10, January 2018*, pages 1–104, Jan 2018.

[23] Joseph C Liberti and Theodore S Rappaport. *Smart antennas for wireless communications: IS-95 and third generation CDMA applications*. Prentice Hall PTR, 1999.

[24] Michel Mouly, Marie-Bernadette Pautet, and Thomas Foreword By-Haug. *The GSM system for mobile communications*. Telecom publishing, 1992.

[25] Andrew J Viterbi and Andrew J Viterbi. *CDMA: principles of spread spectrum communication*, volume 122. Addison-Wesley Reading, MA, 1995.

[26] Qi Bi, Ronald R Brown, Dongzhe Cui, Asif D Gandhi, Ching-Yao Huang, and Stan Vitebsky. Performance of 1xEV-DO third-generation wireless high-speed data systems. *Bell Labs Technical Journal*, 7(3):97–107, 2003.

[27] David Astély, Erik Dahlman, Anders Furuskär, Ylva Jading, Magnus Lindström, and Stefan Parkvall. LTE: the evolution of mobile broadband. *IEEE Communications Magazine*, 47(4), 2009.

[28] Stefan Parkvall, Anders Furuskar, and Erik Dahlman. Evolution of LTE toward IMT-advanced. *IEEE Communications Magazine*, 49(2), 2011.

[29] Jeffrey G Andrews, Stefano Buzzi, Wan Choi, Stephen V Hanly, Angel Lozano, Anthony CK Soong, and Jianzhong Charlie Zhang. What will 5G be? *IEEE Journal on selected areas in communications*, 32(6):1065–1082, 2014.

[30] Asfandyar Qureshi, Jennifer Carlisle, and John Guttag. Tavarua: video streaming with wwan striping. In *Proceedings of the 14th ACM international conference on Multimedia*, pages 327–336. ACM, 2006.

[31] Chris Hurley. *WarDriving: Drive, detect, defend: A guide to wireless security*. Elsevier, 2004.

[32] Martin Heusse, Franck Rousseau, Gilles Berger-Sabbatel, and Andrzej Duda. Performance anomaly of 802.11 b. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 836–843. IEEE, 2003.

[33] Dimitris Vassis, George Kormentzas, Angelos Rouskas, and Ilias Maglogiannis. The IEEE 802.11 g standard for high data rate WLANs. *IEEE network*, 19(3):21–26, 2005.

[34] Yang Xiao. IEEE 802.11 n: enhancements for higher throughput in wireless LANs. *IEEE Wireless Communications*, 12(6):82–91, 2005.

[35] Eng Hwee Ong, Jarkko Kneckt, Olli Alanen, Zheng Chang, Toni Huovinen, and Timo Nihtilä. IEEE 802.11 ac: Enhancements for very high throughput WLANs. In *Personal indoor and mobile radio communications (PIMRC), 2011 IEEE 22nd international symposium on*, pages 849–853. IEEE, 2011.

[36] Eldad Perahia, Carlos Cordeiro, Minyoung Park, and L Lily Yang. IEEE 802.11 ad: Defining the next generation multi-Gbps Wi-Fi. In *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, pages 1–5. IEEE, 2010.

[37] IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput. *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, pages 1–565, Oct 2009.

[38] IEEE Standard for Information technology– Telecommunications and information exchange between systems Local and metropolitan area networks– Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications–Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz. *IEEE Std 802.11ac-2013 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012, IEEE Std 802.11aa-2012, and IEEE Std 802.11ad-2012)*, pages 1–425, Dec 2013.

[39] IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band. *IEEE Std 802.11ad-2012 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012 and IEEE Std 802.11aa-2012)*, pages 1–628, Dec 2012.

[40] Boris Bellalta. IEEE 802.11 ax: High-efficiency WLANs. *IEEE Wireless Communications*, 23(1):38–46, 2016.

[41] D-Link, Asus tout 802.11ax Wi-Fi routers, but you'll have to wait until later in 2018. `https://www.zdnet.com/article/d-link-asus-tout-802-11ax-wi-fi-routers-but-youll-have-to-wait-until-later-in-2018/`. Accessed: 2019-06-01.

[42] 802.11ay - Standard for Information Technology. https://standards.ieee.org/develop/project/802.11ay.html. Accessed: 2019-06-01.

[43] 802.11az - Standard for Information Technology. https://standards.ieee.org/develop/project/802.11az.html. Accessed: 2019-06-01.

[44] Bo Li, Yang Qin, Chor Ping Low, and Choon Lim Gwee. A survey on mobile WiMAX [wireless broadband access]. *IEEE Communications Magazine*, 45(12):70–75, 2007.

[45] Padmanand Warrier and Balaji Kumar. *xDSL Architecture*. McGraw-Hill Companies, 2000.

[46] Seung-Que Lee, Namhun Park, Choongho Cho, Hyongwoo Lee, and Seungwan Ryu. The wireless broadband (wibro) system for broadband wireless internet services. *IEEE Communications Magazine*, 44(7):106–112, 2006.

[47] Mongnam Han, Youngseok Lee, Sue Moon, Keon Jang, and Dooyoung Lee. Evaluation of VoIP quality over WiBro. In *International Conference on Passive and Active Network Measurement*, pages 51–60. Springer, 2008.

[48] Dirk Reichardt, Maurizio Miglietta, Lino Moretti, Peter Morsink, and Wolfgang Schulz. CarTALK 2000: Safe and comfortable driving based upon inter-vehicle-communication. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 545–550. IEEE, 2002.

[49] WJ Franz, Reinhold Eberhardt, and Thomas Luckenbach. Fleetnet-internet on the road. In *8th World Congress on Intelligent Transport Systems ITS America, ITS Australia, ERTICO (Intelligent Transport Systems and Services-Europe)*, 2001.

[50] Jinyang Li, John Jannotti, Douglas SJ De Couto, David R Karger, and Robert Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 120–130. ACM, 2000.

[51] Michael Käsemann, Holger Füßler, Hannes Hartenstein, and Martin Mauve. A reactive location service for mobile ad hoc networks. *Technical reports*, 2, 2002.

[52] Michael Käsemann, Hannes Hartenstein, Holger Füßler, Martin Mauve, et al. Analysis of a Location Service for Position-Based Routing in Mobile Ad Hoc Networks. In *WMAN*, pages 121–133, 2002.

[53] Mario Gerla, Biao Zhou, Yeng-Zhong Lee, Fabio Soldo, Uichin Lee, and Gustavo Marfia. Vehicular grid communications: the role of the internet infrastructure. In *Proceedings of the 2nd annual international workshop on Wireless internet*, page 19. ACM, 2006.

[54] Christian Lochert, Hannes Hartenstein, Jing Tian, Holger Fussler, Dagmar Hermann, and Martin Mauve. A routing strategy for vehicular ad hoc networks in city environments. In *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*, pages 156–161. IEEE, 2003.

[55] David B Johnson, David A Maltz, Josh Broch, et al. DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. *Ad hoc networking*, 5:139–172, 2001.

[56] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. Ad hoc on-demand distance vector (AODV) routing. Technical report, 2003.

[57] Brad Karp and Hsiang-Tsung Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254. ACM, 2000.

[58] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[59] Jack Edmonds. Matroids and the greedy algorithm. *Mathematical programming*, 1(1):127–136, 1971.

[60] Jing Tian, Lu Han, and Kurt Rothermel. Spatially aware packet routing for mobile ad hoc inter-vehicle radio networks. In *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*, volume 2, pages 1546–1551. IEEE, 2003.

[61] Boon-Chong Seet, Genping Liu, Bu-Sung Lee, Chuan-Heng Foh, Kai-Juan Wong, and Keok-Kee Lee. A-STAR: A mobile ad hoc routing strategy for metropolis vehicular communications. In *International Conference on Research in Networking*, pages 989–999. Springer, 2004.

[62] Ljubica Blažević, Silvia Giordano, and Jean-Yves Le Boudec. Self organized terminode routing. *Cluster Computing*, 5(2):205–218, 2002.

[63] Valery Naumov and Thomas R Gross. Connectivity-aware routing (CAR) in vehicular ad-hoc networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1919–1927. IEEE, 2007.

[64] Valery Naumov, Rainer Baumann, and Thomas Gross. An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 108–119. ACM, 2006.

[65] Christian Lochert, Martin Mauve, Holger Füßler, and Hannes Hartenstein. Geographic routing in city scenarios. *ACM SIGMOBILE mobile computing and communications review*, 9(1):69–72, 2005.

[66] Guoming Tang, Yi Xie, Daquan Tang, and Jiuyang Tang. Divisional perimeter routing for GPSR based on left and right hand rules. In *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, volume 2, pages 726–729. IEEE, 2011.

[67] Jing Zhao and Guohong Cao. VADD: Vehicle-assisted data delivery in vehicular ad hoc networks. *IEEE transactions on vehicular technology*, 57(3):1910–1922, 2008.

[68] Pratap Misra and Per Enge. Global positioning system: Signals, measurements and performance second edition. *Massachusetts: Ganga-Jamuna Press*, 2006.

[69] Dragos Niculescu and Badri Nath. Trajectory based forwarding and its applications. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 260–272. ACM, 2003.

[70] Ilias Leontiadis and Cecilia Mascolo. GeOpps: Geographical opportunistic routing for vehicular networks. In *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, pages 1–6. IEEE, 2007.

[71] National Highway Traffic Safety Administration, U.S. Department of Transportation. `https://www.nhtsa.gov/`. Accessed: 2019-06-01.

[72] CAMP Vehicle Safety Communications Consortium et al. Vehicle safety communications project: Task 3 final report: identify intelligent vehicle safety applications enabled by DSRC. *National Highway Traffic Safety Administration, US Department of Transportation, Washington DC*, 2005.

[73] Daniel Jiang, Vikas Taliwal, Andreas Meier, Wieland Holfelder, and Ralf Herrtwich. Design of 5.9 GHz DSRC-based vehicular safety communication. *IEEE Wireless Communications*, 13(5), 2006.

[74] Traffic Signal Timing Manual, Federal Highway Administration (FHWA), U.S. Department of Transportation. `https://ops.fhwa.dot.gov/`

publications/fhwahop08024/chapter9.htm. Accessed: 2019-06-01.

[75] Uichin Lee, Eugenio Magistretti, Mario Gerla, Paolo Bellavista, and Antonio Corradi. Dissemination and harvesting of urban data using vehicular sensing platforms. *IEEE transactions on vehicular technology*, 58(2):882–901, 2009.

[76] Uichin Lee, Eugenio Magistretti, Mario Gerla, Paolo Bellavista, Pietro Lió, and Kang-Won Lee. Bio-inspired multi-agent data harvesting in a proactive urban monitoring environment. *Ad Hoc Networks*, 7(4):725–741, 2009.

[77] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. CarTel: a distributed mobile sensor computing system. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 125–138. ACM, 2006.

[78] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 29–39. ACM, 2008.

[79] Alok Nandan, Shirshanka Das, Giovanni Pau, Mario Gerla, and MY Sanadidi. Co-operative downloading in vehicular ad-hoc wireless networks. In *Wireless On-demand Network Systems and Services, 2005. WONS 2005. Second Annual Conference on*, pages 32–41. IEEE, 2005.

[80] Alok Nandan, Saurabh Tewari, Shirshanka Das, Mario Gerla, and Leonard Kleinrock. AdTorrent: Delivering location cognizant advertisements to car networks. In *WONS 2006: Third Annual Conference on Wireless On-demand Network Systems and Services*, pages 203–212, 2006.

[81] Murat Caliskan, Daniel Graupner, and Martin Mauve. Decentralized discovery of free parking places. In *Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, pages 30–39. ACM, 2006.

[82] Tamer Nadeem, Sasan Dashtinezhad, Chunyuan Liao, and Liviu Iftode. TrafficView: traffic data dissemination using car-to-car communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 8(3):6–19, 2004.

[83] Uichin Lee, Jiyeon Lee, Joon-Sang Park, and Mario Gerla. FleaNet: A virtual market place on vehicular networks. *IEEE Transactions on Vehicular Technology*, 59(1):344–355, 2010.

[84] Meng Guo, Mostafa H Ammar, and Ellen W Zegura. V3: A vehicle-to-vehicle live video streaming architecture. *Pervasive and Mobile Computing*, 1(4):404–424, 2005.

[85] Jungkeun Yoon, Brian Noble, and Mingyan Liu. Surface street traffic estimation. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 220–232. ACM, 2007.

[86] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. *ACM SIGARCH Computer Architecture News*, 30(5):96–107, 2002.

[87] Oriana Riva and Cristian Borcea. The urbanet revolution: Sensor power to the people! *IEEE Pervasive Computing*, 6(2), 2007.

[88] Shane B Eisenman, Nicholas D Lane, Emiliano Miluzzo, Ronald A Peterson, Gahng-Seop Ahn, and Andrew Campbell. Metrosense project: People-centric sensing at scale. In *Workshop on World-Sensor-Web (WSW 2006), Boulder*. Citeseer, 2006.

[89] Rahul C Shah, Sumit Roy, Sushant Jain, and Waylon Brunette. Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks. *Ad Hoc Networks*, 1(2-3):215–233, 2003.

[90] Johan Pouwelse, Paweł Garbacki, Dick Epema, and Henk Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *International Workshop on Peer-to-Peer Systems*, pages 205–216. Springer, 2005.

[91] Kevin C Lee, Seung-Hoon Lee, Ryan Cheung, Uichin Lee, and Mario Gerla. First experience with cartorrent in a real vehicular ad hoc network testbed. In *2007 Mobile Networking for Vehicular Environments*, pages 109–114. IEEE, 2007.

[92] Uichin Lee, Joon-Sang Park, Joseph Yeh, Giovanni Pau, and Mario Gerla. Code torrent: content distribution using network coding in VANET. In *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, pages 1–5. ACM, 2006.

[93] Joon-Sang Park, Uichin Lee, Soon Y Oh, Mario Gerla, and Desmond S Lun. Emergency related video streaming in VANET using network coding. In *Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, pages 102–103. ACM, 2006.

[94] Marios D Dikaiakos, Saif Iqbal, Tamer Nadeem, and Liviu Iftode. VITP: an information transfer protocol for vehicular computing. In *Proceedings of the 2nd ACM international workshop on Vehicular ad hoc networks*, pages 30–39. ACM, 2005.

[95] Jedrzej Rybicki, Björn Scheuermann, Wolfgang Kiess, Christian Lochert, Pezhman Fallahi, and Martin Mauve. Challenge: peers on wheels-a road to new traffic information systems. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 215–221. ACM, 2007.

[96] Stephen Smaldone, Lu Han, Pravin Shankar, and Liviu Iftode. RoadSpeak: enabling voice chat on roadways using vehicular social networks. In *Proceedings of the 1st Workshop on Social Network Systems*, pages 43–48. ACM, 2008.

[97] Victor Gradinescu, Cristian Gorgorin, Raluca Diaconescu, Valentin Cristea, and Liviu Iftode. Adaptive traffic lights using car-to-car communication. In *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, pages 21–25. IEEE, 2007.

[98] Peng Zhou, Tamer Nadeem, Porlin Kang, Cristian Borcea, and Liviu Iftode. EZCab: A cab booking application using short-range wireless communication. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 27–38. IEEE, 2005.

[99] Elgan Huang, Wenjun Hu, Jon Crowcroft, and Ian Wassell. Towards commercial mobile ad hoc network applications: A radio dispatch system. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 355–365. ACM, 2005.

[100] Claudio E Palazzi, Marco Roccetti, Stefano Ferretti, Giovanni Pau, and Mario Gerla. Online games on wheels: Fast game event delivery in vehicular ad-hoc networks. *Proc. of V2VCOM*, 2007.

[101] John Whipple, William Arensman, and Marian Starr Boler. A public safety application of GPS-enabled smartphones and the android operating system. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 2059–2061. IEEE, 2009.

[102] Ming-Chiao Chen, Jiann-Liang Chen, and Teng-Wen Chang. Android/OSGi-based vehicular network management system. *Computer Communications*, 34(2):169–183, 2011.

[103] Jie Yang, Jie Wang, and Benyuan Liu. An intersection collision warning system using Wi-Fi smartphones in VANET. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–5. IEEE, 2011.

[104] International Organization for Standardization. ISO 14230-1:1999: Road vehicles, Diagnostic systems, Keyword protocol 2000. 1999.

[105] Jorge Zaldivar, Carlos T Calafate, Juan-Carlos Cano, and Pietro Manzoni. Providing accident detection in vehicular networks through OBD-II devices and Android-based smartphones. In *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, pages 813–819. IEEE, 2011.

[106] Bojin Liu, Dipak Ghosal, Yachao Dong, Chen-Nee Chuah, and Michael Zhang. CarbonRecorder: A Mobile-Social Vehicular Carbon Emission Tracking Application Suite. In *Vehicular Technology Conference (VTC Fall), 2011 IEEE*, pages 1–2. IEEE, 2011.

[107] Stefan Diewald, Andreas Möller, Luis Roalter, and Matthias Kranz. DriveAssist-A V2X-Based Driver Assistance System for Android. In *Mensch & Computer Workshopband*, pages 373–380, 2012.

[108] Claudia Campolo, Antonio Iera, Antonella Molinaro, Stefano Yuri Paratore, and Giuseppe Ruggeri. SMaRTCaR: An integrated smartphone-based platform to support traffic management applications. In *Vehicular Traffic Management for Smart Cities (VTM), 2012 First International Workshop on*, pages 1–6. IEEE, 2012.

[109] Johan Wideberg, Pablo Luque, and Daniel Mantaras. A smartphone application to extract safety and environmental related information from the OBD-II interface of a car. *International Journal of Vehicle Systems Modelling and Testing*, 7(1):1–11, 2012.

[110] Pedro Emanuel Rodrigues Gomes, Fausto Vieira, and Michel Ferreira. The See-Through System: From implementation to test-drive. In *VNC*, pages 40–47, 2012.

[111] Emmanouil Koukoumidis, Margaret Martonosi, and Li-Shiuan Peh. Leveraging smartphone cameras for collaborative road advisories. *Mobile Computing, IEEE Transactions on*, 11(5):707–723, 2012.

[112] Chuang-Wen You, Nicholas D Lane, Fanglin Chen, Rui Wang, Zhenyu Chen, Thomas J Bao, Martha Montes-de Oca, Yuting Cheng, Mu Lin, Lorenzo Torresani, et al. Carsafe app: Alerting drowsy and distracted drivers using dual cameras on smartphones. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 13–26. ACM, 2013.

[113] Javier E Meseguer, Carlos T Calafate, Juan Carlos Cano, and Pietro Manzoni. Characterizing the Driving Style Behavior using Artificial Intelligence Techniques.

[114] Waze official website. `https://www.waze.com/`. Accessed: 2019-06-01.

[115] Torque official website. `http://torque-bhp.com/`. Accessed: 2019-06-01.

[116] iOnRoad official website. `http://www.ionroad.com/`. Accessed: 2018-04-28.

[117] ETSI official website. `http://www.etsi.org/`. Accessed: 2019-06-01.

[118] CEN Technical Committee 278. `http://www.itsstandards.eu/`. Accessed: 2019-06-01.

[119] ISO Technical Committee 204. `https://www.iso.org/committee/54706.html`. Accessed: 2019-06-01.

[120] Stephen E Deering. Internet protocol, version 6 (IPv6) specification. 1998.

[121] David Johnson, Charles Perkins, and Jari Arkko. Mobility support in IPv6. Technical report, 2004.

[122] Thierry Ernst, Koshiro Mitsuya, and Keisuke Uehara. Network Mobility from the InternetCAR Perspective. *Journal of Interconnection Networks*, 04(03):329–343, 2003.

[123] M Wolf, Tim Leinmüller, Alexandre Petrescu, Christophe Janneteau, H Lach, Christoph Barz, M Frank, M Pilz, Miklós Aurél Rónai, Kristóf Fodor, and Ralf Tönjes. IPv6 based OverDRiVE Moving Networks for Vehicles. 52, 11 2005.

[124] Ralf Tönjes, Klaus Moessner, Thorsten Lohmar, and Michael Wolf. OverDRiVE–Spectrum Efficient Multicast Services to Vehicles. 2002.

[125] Lin Xu, Ralf Tonjes, Toni Paila, Wolfgang Hansmann, Matthias Frank, and Markus Albrecht. DRiVE-ing to the Internet: Dynamic radio for IP services in vehicular environments. In *Local Computer Networks, 2000. LCN 2000. Proceedings. 25th Annual IEEE Conference on*, pages 281–289. IEEE, 2000.

[126] Gwenaelle Toulminet, Jacques Boussuge, and Claude Laurgeau. Comparative synthesis of the 3 main European projects dealing with Cooperative Systems (CVIS, SAFESPOT and COOPERS) and description of COOPERS Demonstration Site 4. In *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, pages 809–814. IEEE, 2008.

[127] Wern-Yarng Shieh, Wei-Hsun Lee, Shen-Lung Tung, Bor-Shenn Jeng, and Cheng-Hsin Liu. Analysis of the optimum configuration of roadside units and onboard units in dedicated short-range communication systems. *IEEE Transactions on Intelligent Transportation Systems*, 7(4):565–571, 2006.

[128] MirrorLink official website. `https://mirrorlink.com/`. Accessed: 2019-06-01.

[129] Android Auto official website. `https://www.android.com/auto/`. Accessed: 2019-06-01.

[130] About Google. `https://www.google.org/`. Accessed: 2019-06-01.

[131] CarPlay official website. `https://www.apple.com/ios/carplay/`. Accessed: 2019-06-01.

[132] Jon Postel. Transmission control protocol. 1981.

[133] Jon Postel. User datagram protocol. Technical report, 1980.

[134] Bur Goode. Voice over internet protocol (VoIP). *Proceedings of the IEEE*, 90(9):1495–1517, 2002.

[135] Matt Richardson and Shawn Wallace. *Getting Started with Raspberry Pi*. O'Reilly Media, Inc., 2012.

[136] William Harrington. *Learning Raspbian*. Packt Publishing Ltd, 2015.

[137] Jose Dieguez Castro. *Introducing Linux Distros*. Apress, Berkeley, CA, 2016.

[138] Dnsmasq documentation. `http://thekelleys.org.uk/dnsmasq/doc.html`. Accessed: 2019-06-01.

[139] Youngsong Mun and Hyewon K Lee. Domain Name System (DNS). *Understanding IPv6*, pages 151–172, 2005.

[140] R Droms. Dynamic Host Configuration Protocol. 1997.

[141] George Tsirtsis and Pyda Srisuresh. Network address translation-protocol translation (NAT-PT). Technical report, 2000.

[142] Iptables project website. http://www.netfilter.org/projects/iptables/. Accessed: 2019-06-01.

[143] Nmcli documentation. https://developer.gnome.org/NetworkManager/stable/nmcli.html. Accessed: 2019-06-01.

[144] NetworkManager documentation. https://developer.gnome.org/NetworkManager/stable/NetworkManager.html. Accessed: 2019-06-01.

[145] Roy T. Fielding and Richard N. Taylor. Principled Design of the Modern Web Architecture. *ACM Trans. Internet Technol.*, 2(2):115–150, May 2002.

[146] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–HTTP/1.1. Technical report, 1999.

[147] The Restlet Framework. https://restlet.com/. Accessed: 2019-06-01.

[148] RockSaw official website. https://www.savarese.com/software/rocksaw/. Accessed: 2019-06-01.

[149] Java official website. https://www.java.com/en/. Accessed: 2018-06-01.

[150] Android official website. https://www.android.com/. Accessed: 2019-06-01.

[151] D-Bus official website. https://www.freedesktop.org/wiki/Software/dbus/. Accessed: 2019-06-01.

[152] Lollipop official website. https://www.android.com/versions/lollipop-5-0/. Accessed: 2019-06-02.

[153] Joshua J Drake, Zach Lanier, Collin Mulliner, Pau Oliva Fora, Stephen A Ridley, and Georg Wicherski. *Android hacker's handbook*. John Wiley & Sons, 2014.

[154] ART. https://source.android.com/devices/tech/dalvik/. Accessed: 2019-06-02.

[155] Dalvik vs ART. https://www.androidpolice.com/2013/11/12/meet-art-part-2-benchmarks-performance-wont-blow-away-today-will-get-better/. Accessed: 2019-06-02.

[156] OsmAnd official website. `https://osmand.net/`. Accessed: 2019-06-02.

[157] OpenStreetMap website. `https://www.openstreetmap.org/`. Accessed: 2019-06-02.

[158] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[159] Jerry D Woll. VORAD collision warning radar. In *Radar conference, 1995., Record of the IEEE 1995 International*, pages 369–372. IEEE, 1995.

[160] General Motors official website. `https://www.gm.com/`. Accessed: 2019-06-02.

[161] Shih-Ken Chen and Jayendra S Parikh. Developing a forward collision warning system simulation. In *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pages 338–343. IEEE, 2000.

[162] Ronald Miller and Qingfeng Huang. An adaptive peer-to-peer collision warning system. In *Vehicular technology conference, 2002. VTC Spring 2002. IEEE 55th*, volume 1, pages 317–321. IEEE, 2002.

[163] Narayan Srinivasa. Vision-based vehicle detection and tracking method for forward collision warning in automobiles. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 626–631. IEEE, 2002.

[164] Narayan Srinivasa, Yang Chen, and Cindy Daniell. A fusion system for real-time forward collision warning in automobiles. In *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*, volume 1, pages 457–462. IEEE, 2003.

[165] Erez Dagan, Ofer Mano, Gideon P Stein, and Amnon Shashua. Forward collision warning with a single camera. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 37–42. IEEE, 2004.

[166] Yan Zhang, Stephen J Kiselewich, and William A Bauson. Legendre and Gabor moments for vehicle recognition in forward collision warning. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pages 1185–1190. IEEE, 2006.

[167] Huei-Yung Lin, Li-Qi Chen, Yu-Hsiang Lin, and Meng-Shiun Yu. Lane departure and front collision warning using a single camera. In *Intelligent Signal Processing and Communications Systems (ISPACS), 2012 International Symposium on*, pages 64–69. IEEE, 2012.

[168] Erik Coelingh, Lotta Jakobsson, Henrik Lind, and Magdalena Lindman. Collision warning with auto brake: a real-life safety perspective. *Innovations for Safety: Opportunities and Challenges*, 2007.

[169] Erik Coelingh, Andreas Eidehall, and Mattias Bengtsson. Collision Warning with Full Auto Brake and Pedestrian Detection - a practical example of Automatic Emergency Braking. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 155–160. IEEE, 2010.

[170] Volvo official website. `https://www.volvocars.com/`. Accessed: 2019-06-02.

[171] O Rebecca Vincent and Olusegun Folorunso. A descriptive algorithm for sobel image edge detection. In *Proceedings of Informing Science & IT Education Conference (InSITE)*, volume 40, pages 97–107. Informing Science Institute California, 2009.

[172] Jing-Fu Liu, Yi-Feng Su, Ming-Kuan Ko, and Pen-Ning Yu. Development of a vision-based driver assistance system with lane departure warning and forward collision warning functions. In *Computing: Techniques and Applications, 2008. DICTA'08. Digital Image*, pages 480–485. IEEE, 2008.

[173] Bao Rong Chang, Hsiu Fen Tsai, and Chung-Ping Young. Intelligent data fusion system for predicting vehicle collision warning using vision/GPS sensing. *Expert Systems with Applications*, 37(3):2439–2450, 2010.

[174] James A Misener, Raja Sengupta, and Hariharan Krishnan. Cooperative collision warning: Enabling crash avoidance with wireless technology. In *12th World Congress on ITS*, volume 3, 2005.

[175] Han-Shue Tan and Jihua Huang. DGPS-based vehicle-to-vehicle cooperative collision warning: Engineering feasibility viewpoints. *IEEE Transactions on Intelligent Transportation Systems*, 7(4):415–428, 2006.

[176] Zhang Lei, Wang Jianqiang, and Li Keqiang. Forward Collision Warning System Based on THASV-II Platform. In *Vehicular Electronics and Safety, 2006. ICVES 2006. IEEE International Conference on*, pages 255–258. IEEE, 2006.

[177] Jaemin Chun, Sung H Han, Gunhyuk Park, Jongman Seo, Seungmoon Choi, et al. Evaluation of vibrotactile feedback for forward collision warning on the steering wheel and seatbelt. *International Journal of Industrial Ergonomics*, 42(5):443–448, 2012.

[178] OpenALPR official website. `https://www.openalpr.com/`. Accessed: 2019-06-02.

[179] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–2041, 2006.

[180] Christian Wolf, J-M Jolion, and Francoise Chassaing. Text localization, enhancement and binarization in multimedia documents. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 2, pages 1037–1040. IEEE, 2002.

[181] Jaakko Sauvola and Matti Pietikäinen. Adaptive document image binarization. *Pattern recognition*, 33(2):225–236, 2000.

[182] Richard O Duda and Peter E Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[183] Road Safety Authority of Ireland suggest the use of two second rule. `http://www.rotr.ie/Rules_of_the_road.pdf`. Accessed: 2019-06-02.

[184] New York State Department of Motor Vehicles. `https://dmv.ny.gov/about-dmv/chapter-8-defensive-driving`. Accessed: 2019-06-02.

[185] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.

[186] Greg Welch and Gary Bishop. An introduction to the Kalman filter. 1995.

[187] National Highway Traffic Safety Administration et al. National motor vehicle crash causation survey: Report to congress. *National Highway Traffic Safety Administration Technical Report DOT HS*, 811:059, 2008.

[188] JA Groeger and BA Clegg. Why isn't driver training contributing more to road safety? In *Behavioural Research in Road Safety IV. Proceedings of a seminar held 6-7 September 1993, Brunel University.(TRL published article PA 3035/94)*, 1994.

[189] David D Clarke, PJ Ward, and J Jones. *Overtaking accidents*. Transport Research Laboratory, 1998.

[190] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the H. 264/AVC video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560–576, 2003.

[191] Gregory K Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.

[192] Sanford Weisberg. *Applied linear regression*, volume 528. John Wiley & Sons, 2005.

[193] Henning Schulzrinne. Real time streaming protocol (RTSP). 1998.

[194] Karel Rijkse. H. 263: video coding for low-bit-rate communication. *IEEE Communications magazine*, 34(12):42–45, 1996.

[195] Greg Roelofs and Richard Koman. *PNG: the definitive guide*. O'Reilly & Associates, Inc., 1999.

[196] Quan Huynh-Thu and Mohammed Ghanbari. Scope of validity of PSNR in image/video quality assessment. *Electronics letters*, 44(13):800–801, 2008.

[197] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[198] Alain Hore and Djemel Ziou. Image quality metrics: PSNR vs. SSIM. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2366–2369. IEEE, 2010.

[199] Zhou Wang and Alan C Bovik. Mean squared error: love it or leave it? A new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.

[200] A Crawford. The overtaking driver. *Ergonomics*, 6(2):153–170, 1963.

[201] Brian L Hills. Vision, visibility, and perception in driving. *Perception*, 9(2):183–216, 1980.

[202] Robert F Phillips. Least absolute deviations estimation via the EM algorithm. *Statistics and Computing*, 12(3):281–285, 2002.

[203] Peter J Rousseeuw and Katrien Van Driessen. Computing LTS regression for large data sets. *Data mining and knowledge discovery*, 12(1):29–45, 2006.

[204] Andrew F Siegel. Robust regression using repeated medians. *Biometrika*, 69(1):242–244, 1982.

[205] Michael G Akritas, Susan A Murphy, and Michael P Lavalley. The Theil-Sen estimator with doubly censored data and applications to astronomy. *Journal of the American Statistical Association*, 90(429):170–177, 1995.

[206] Jelena Mirkovic, Sven Dietrich, David Dittrich, and Peter Reiher. *Internet Denial of Service: Attack and Defense Mechanisms (Radia Perlman Computer Networking and Security).* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[207] S. Chen, J. Xu, E. Sezer, P. Gauriar, and R. Iyer. Non-control-data attacks are realistic threats. In *Proceedings of the Usenix Security Symposium*, pages 177–192, 2005.

[208] Chun-Kan Fung and M. C. Lee. A denial-of-service resistant public-key authentication and key establishment protocol. In *Conference Proceedings of the IEEE International Performance, Computing, and Communications Conference (Cat. No.02CH37326)*, pages 171–178, 2002.

[209] Android-Data-Storage. `https://developer.android.com/guide/topics/data/data-storage.html`. Accessed: 2019-06-02.

[210] Martín Abadi, Mihai Budiu, Úlfar Erlingsson, and Jay Ligatti. Control-flow integrity. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, CCS ’05, pages 340–353, New York, NY, USA, 2005. ACM.